CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA

PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA

CURSO SUPERIOR DE TECNOLOGIA EM REDES

RICARDO BASTOS GARCIA

SISTEMA DE DETECÇÃO DE INTRUSÃO E BLOQUEIO DE ATAQUES UTILIZANDO IDS-SNORT

LINS/SP

1º SEMESTRE/2013

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA

CURSO SUPERIOR DE TECNOLOGIA EM REDES

RICARDO BASTOS GARCIA

SISTEMA DE DETECÇÃO DE INTRUSÃO E BLOQUEIO DE ATAQUES UTILIZANDO IDS-SNORT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins, para obtenção de título de tecnólogo em Redes de Computadores

Orientador: Prof. Dr. Fernando A. Garcia Muzzi

LINS/SP

1º SEMESTRE/2013

RICARDO BASTOS GARCIA

SISTEMA DE DETECÇÃO DE INTRUSÃO E BLOQUEIO DE ATAQUES UTILIZANDO IDS-SNORT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins Prof. Antonio Seabra, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Redes de Computadores sob orientação do Prof. Dr. Fernando A. Garcia Muzzi

Data de aprovação: ___/___/___

____

Orientador (Prof. Dr. Fernando A. Garcia Muzzi)

____

Examinador 1 (Nome do Examinador)

____

Examinador 2 (Nome do Examinador)

Dedicatória

Antes de tudo, dedico este trabalho a Deus que conduziu meus caminhos na superação dos momentos mais difíceis. A meus pais, Deoclécio Garcia Fernandes, *in memorian*, e Maria Inez Bastos Garcia, *in memorian*, e a minha amada filha Julia.

Ricardo Bastos Garcia

AGRADECIMENTOS

Neste momento tão importante pra mim, eu não poderia deixar de agradecer às pessoas de que, de alguma maneira contribuíram para que este momento acontecesse.

Agradeço em especial ao meu orientador, Prof. Dr. Fernando A. Garcia Muzzi, por quem tive a honra de ser orientado, assim também, pelas orientações proveitosas e competentes, e acima de tudo, por nos fazer acreditar que este trabalho sempre seria possível.

Ao professor Ygor Gonzaga de Oliveira, que não está mais entre nós, e cuja amizade muito contribuiu em grandeza de espírito.

Aos professores Alexandre Ponce, Adriana de Bortoli e Luciane Noronha pela correção do nosso trabalho e pela disposição para tirar as dúvidas.

Ao professor Julio Fernando Lieira pela ajuda e pelo material cedido, que muito me ajudou na construção dos principais capítulos.

A todos os demais professores agradeço pela dedicação a aulas ministradas.

E aos amigos de turma pelos momentos em que compartilhamos conhecimentos, companheirismo, amizade, dificuldades sempre superadas e alegrias.

Ricardo Bastos Garcia

RESUMO

A tecnologia da informação tem evoluído rapidamente. Computadores mais avançados, redes mais velozes e abrangentes, softwares inteligentes de gestão, tudo para auxiliar nas mais diversas atividades. Porém, ao mesmo tempo, existem alguns que preferem fazer uso de toda esta tecnologia para causar algum dano a outrem. Assim surgiram os chamados “hackers”. Com o propósito de prevenir a ação destes indivíduos, criar uma política de segurança é de vital importância para proteger as informações. Segurança é a direção em que se pode viajar, mas que de fato, nunca se chegar ao destino. É um processo dinâmico de constante acompanhamento, aprendizado e melhoria. Muitas são as ferramentas usadas na configuração de um sistema de segurança de redes. Entre estas ferramentas estão os IDS. Os IDS são sistemas de detecção de intrusão que monitoram as atividades da rede procurando por padrões de comportamento que se enquadre em tentativas conhecidas de acessos não autorizados. Nesse contexto, foi usado como base ao longo deste trabalho o IDS *Snort.* É utilizada para análise uma plataforma de simulação, utilizando-se máquinas virtuais, instalando e configurando as ferramentas necessárias para criar um ambiente para testes. As principais contribuições são a detecção e contenção das tentativas de invasão e avaliação dos resultados para construção de conhecimento da segurança da rede interna.

Palavras-chave : Segurança, IDS, Ataques, Contenção.

ABSTRACT

Information technology has evolved rapidly. Most advanced computers, faster networks and comprehensive, intelligent management software, all to assist in various activities. However, while there are some who prefer to make use of all this technology to cause harm to others. Thus appeared the so-called "hackers". In order to prevent the action of these individuals, creating a security policy is vitally important to protect the information. Security is the direction in which they can travel, but never actually reach the destination. It is a dynamic process of constant monitoring, learning and improvement. Many tools are used in the configuration of a system of network security. Among these tools are the IDS. The IDS are intrusion detection systems that monitor the activities of the network looking for patterns of behavior that fits known attempts at unauthorized access. In this context, we use as the basis of this work over the Snort IDS. It is a platform used for analysis, simulation using virtual machines, installing and configuring the tools necessary to create an environment for testing. The main contributions are the detection and containment of intrusion attempts and evaluating results to build knowledge of internal network security.

Keywords: Security, IDS, Attacks, Containment.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACID – *Analysis Console for Intrusion Databases*
DMZ – *DeMilitarized Zone*
HD – *Hard Disk*
HIDS – *Host-Based Intrusion Detection System*
HTTP – *Hyper Text Transfer Protocol*
ICMP – *Internet Control Message Protocol*
IDS – *Intrusion Detection System*
IMAP – *Internet Message Access Protocol*
IP – *Internet Protocol*
IPS – *Intrusion Prevention System*
IPSEC– *IP Security Protocol*
IPX – *Internetwork Packet Exchange*
ISP – *Internet Service Provider*
LAN – *Local Area Network*
NIC – *Network Interface Card*
NIDS – *Network-Based Intrusion Detection System*
POP – *Point Of Presence*
RPC – *Remote procedure Call*
SIP – *Session Initiation Protocol*
SMTP – *Simple Mail Transfer Protocol*
SNMP – *Simple Network Management Protocol*
SPAN – *Switched Port Analyzer*
SPX – *Sequenced Packet Exchange*
SQL – *Structured Query Language*
SO – *Sistema Operacional*
TCP – *Transmission Control Protocol*
URL – *Uniform Resource Locator*
VPN – *Virtual Private Network*

LISTA DE SÍMBOLOS

@ - Arroba

™ - Marca Registrada

® - Registado

SUMÁRIO

INTRODUÇÃO

Uma revolução está em curso, na qual a informação vale muito. As redes de computadores necessitam cada vez mais atenção voltada a sua segurança, pois muitas são as tecnologias envolvidas, e estas, em constante evolução. A possibilidade de acesso não autorizado, interno ou externo, põe em risco a integridade das informações contidas nestas redes.

> Entende-se por integridade a informação que não foi manipulada sem autorização, isto é, um arquivo pode ser removido ou alterado normalmente com autorização devida, porém, caso um agente externo o faça, esta informação está com sua integridade comprometida. (LARI; AMARAL, 2004, p.5).

Criar um sistema seguro significa anular ou minimizar as chances de um invasor conseguir obter acesso a ele. Assim como as tecnologias envolvidas, as tentativas para burlar a segurança também estão em constante evolução. Para que administradores de redes possam agir no sentido de impossibilitar ou mesmo conter estas tentativas de intrusão, estes devem, primeiramente, tomar conhecimento desta possível movimentação não autorizada agindo em suas redes.

Neste contexto, existem técnicas e ferramentas que são usadas pelos administradores de rede para monitorar a movimentação em suas redes. Dentre as ferramentas estão os sistemas de detecção de intrusão, que tem suas raízes nos sistemas de auditoria financeira, e cuja ideia básica é o monitoramento das atividades do tráfego da rede, reportando assim ao administrador, situações anômalas que possam ocorrer.

Sendo a prevenção o melhor método contra a invasão de sistemas, este trabalho tem como objetivo geral mostrar o funcionamento de uma ferramenta de detecção de intrusão implementada em sistema operacional Linux e configurado para fazer contenção (bloqueio) de ataques DDoS.

Como ferramenta IDS foi adotada o *Snort*. O IDS é muito difundido no mundo do *software* livre e apesar de sua distribuição gratuita, é uma ferramenta poderosa.

Para alcançar o objetivo geral foram buscados os seguintes objetivos específicos:

- Instalar e configurar o *Snort* para que faça o monitoramento das

atividades de um host num ambiente de teste e, utilizando seu conjunto pré-definido de regras, reporte as tentativas de intrusão, bem como, as utilize na geração de regras no *firewall*;

- Para adicionar funcionalidade ao *Snort*, instalar e configurar ferramentas auxiliares, sendo estas : Banco de Dados MySQL, servidor Web Apache, Linguagem PHP5, analisador de tráfego BASE, atualizador automático de regras para o *Snort* OINKMASTER e o GUARDIAN, que age no que se refere a contenção de ataques, gerando regras no *firewall* do sistema;
- Criar ambiente para simulação fazendo uso de máquinas virtuais criadas com o *software* VitualBox;
- Simular um ataque do tipo DDoS utilizando injetores de pacotes;
- Fazer análise das tentativas de intrusão reportadas pelo Snort em um *host* sob ataque DDoS e comparando-as com o mesmo cenário, porém com o modo de contenção (bloqueio) ativado;
- Avaliar as informações levantadas na simulação.

Diante disto, este trabalho se justifica como forma de avaliar a utilização de um sistema de detecção de intrusão de distribuição gratuita, complementando o uso do *firewall* do sistema operacional como mecanismos de segurança indispensáveis.

Inicialmente será feita uma explanação do conceito de IDS, seu funcionamento e classificação de intrusões, quais os tipos de monitoramento, vantagens e desvantagens, apresentados no primeiro capítulo.

No capítulo 2 é apresentado o SNORT, ferramenta IDS com licença “open-source”, indicada para monitorar redes TCP/IP e usada para detectar tráfego suspeito, assim como ataques externos.

O capítulo 3 mostra de forma resumida as principais formas de ataques e invasões e as vulnerabilidades que estas exploram, bem como, o ataque do tipo DDoS que será utilizado na simulação de ataque ao *host*.

No capítulo 4 é mostrado o desenvolvimento do trabalho, as ferramentas utilizadas em conjunto com o *Snort* afim de que este adquira as funcionalidades necessárias para a realização do objetivo do trabalho, bem como as demais ferramentas utilizadas para criarmos as métricas para avaliação.

Finalmente no capítulo 5 são discutidos os resultados experimentais,

apresentados os gráficos estatísticos, quadros comparativos, assim como as conclusões referentes aos testes de avaliação e desempenho e conclusões finais.

1 IDS

O objetivo deste capítulo é mostrar os conceitos que envolvem a segurança dos sistemas nas redes de computadores. Em seguida, apresentar a definição do que é um *Intrusion Detection System* (IDS), abordando seus aspectos, princípios de prevenção, tipos de atuação, metodologias de detecção, vantagens e deficiências, assim como sua importância e contribuição no segmento das tecnologias de segurança da informação.

1.1 SEGURANÇA

A questão chave em redes de computadores é o compartilhamento de recursos, em especial o de dados, que precisam estar ao alcance de todos os usuários da rede independentemente de sua localização física. A internet proporcionou essa interligação em nível mundial, ligando computadores de empresas, indivíduos, dispositivos móveis e usuários domésticos, permitindo o acesso de qualquer ponto para qualquer ponto do planeta. Porém, ao mesmo tempo em que a tecnologia disponibiliza esse poderoso recurso, também remete ao complexo tema voltado a segurança de sistemas nas redes de computadores.

Segundo Lieira (2012), quando se analisa a segurança em sistemas de informação, é necessário ter em mente duas premissas:. Primeiro, a segurança não é uma tecnologia. Não é um dispositivo que se possa comprar e que torne uma rede 100% segura, assim como não é possível também, comprar ou criar um *software* capaz de tornar um computador 100% seguro. E segundo, a segurança não é um estado que se pode atingir. A segurança é uma direção em que se pode viajar, porém, nunca se chegar de fato ao destino. O que se pode fazer é administrar um nível aceitável de risco.

Ainda de acordo com Lieira (2012), outro aspecto de segurança é o fato desta não ser estática, ou seja, é um trabalho de constante atualização, redefinição e implementação. Usando como analogia o ato de tentar subir uma escada rolante infinita que desce. Pode-se correr alguns degraus acima e então parar para recuperar o fôlego; após descansar, descobrirá que a escada rolante o levou para baixo ou até o mesmo para além do ponto de início. Para se manter no mesmo nível, é preciso manter um esforço contínuo. E para avançar e conseguir maior segurança,

será necessário um esforço ainda maior. Como em uma escada rolante, não basta a quantidade do esforço; o esforço precisará ser realizado na direção correta. A pergunta que surge então é, como caminhar na direção correta? A resposta a essa pergunta é, através de uma Política de Segurança. Ou seja, traçar regras do que pode e do que não pode ser feito nos sistemas de informação da empresa, assim como conscientizar os usuários desses sistemas, da necessidade de seguir essas regras.

> A tarefa de prover segurança a um sistema começa com a elaboração de uma política de segurança, onde fatores humanos e gerenciais são considerados. Isso quer dizer que a implementação da segurança depende das necessidades de quem utiliza o sistema. (LARI e AMARAL, 2004, p.5)

Conforme Lari e Amaral (2004), um exemplo comum, é o uso da porta 21 utilizada pelo *File Transfer Protocol* (FTP). Esse protocolo possibilita a troca de arquivos dentro da rede, porém é, ao mesmo tempo, uma brecha de segurança para invasão. Sempre que o *firewall* bloqueia essa porta, evitando assim ataques a mesma, os usuários não poderão enviar arquivos por meio dela para outra máquina da rede interna. Logo, esse tipo de bloqueio deve ser avaliado na política de segurança.

Segundo Lari e Amaral (2004), avaliar a segurança de uma rede remete a questões como, quanto vale a informação e quanto pagar para protegê-las em uma empresa. Responde-se a isto afirmando que o valor que se dá a informação é proporcional a quanto se gasta em protegê-la. Esta é uma tendência mundial, pois os gastos com segurança crescem e se fazem cada vez mais necessários à medida que novos ataques vão surgindo.

1.2 AMEAÇAS E ATAQUES

De acordo com Nobre (2007), uma ameaça, consiste em uma possível violação de um sistema computacional, que pode ser acidental ou intencional. Uma ameaça acidental é aquela que não foi planejada, podendo ser, por exemplo, uma falha no *hardware* ou no *software*. Já uma ameaça intencional está associada à intencionalidade premeditada. Podendo ser desde um monitoramento não autorizado do sistema até ataques sofisticados, como os realizados por *Hackers*.

Ainda de acordo com Nobre (2007), algumas das principais ameaças aos

sistemas nas redes de computadores envolvem destruição ou modificação de informações, roubo, remoção ou perda de informação, revelação de dados confidenciais ou não, e em casos extremos, chegando até a paralisação dos serviços de rede.

Um ataque acontece quando se efetiva uma ameaça intencional. Estes ocorrem por vários motivos. Variam desde a pura curiosidade, interesse em adquirir maior conhecimento sobre os sistemas, intenção em conseguir ganhos financeiros, extorsão, chantagem de algum tipo, espionagem industrial ou venda de informações confidenciais. Outro tipo de interesse é o de ferir a imagem de um governo ou uma determinada empresa ou serviço, e quando isso acontece, a notícia da invasão é proporcional à fama de quem a sofreu e normalmente representa um desastre em termos de repercussão pública. (NOBRE, 2007)

Segundo Nobre (2007), consideram-se três aspectos básicos que um sistema de segurança da informação deve atender para evitar a concretização das ameaças: Prevenção, Detecção e Recuperação. O primeiro deles, a Prevenção, subdividi-se em três tipos distintos, que são primeiramente a proteção de *hardware*, normalmente chamada de segurança física, que impede acessos físicos não autorizados à infra-estrutura da rede, prevenindo roubos de dados, desligamento de equipamentos e demais danos quando se está fisicamente no local. O segundo, é a proteção de arquivos e dados proporcionada pela autenticação, controle de acesso e sistemas antivírus, sendo que, no processo de autenticação é verificada a identidade do usuário, que disponibiliza apenas as transações pertinentes ao mesmo e os programas antivírus garantem a proteção do sistema contra programas maliciosos. E o terceiro tipo de prevenção, a proteção de perímetro, que se encontra nas ferramentas de *firewall* e *routers* que cuidam desse aspecto, mantendo a rede protegida contra tentativas de intrusão interna e externa à rede.

O segundo aspecto, a detecção, que subdividi-se em dois tipos. O primeiro são os alertas no qual os sistemas de detecção de intrusões alertam os responsáveis pela segurança sobre qualquer sinal de invasão ou mudança suspeita no comportamento da rede que possa significar um padrão de ataque. Os avisos podem ser via e-mail, mensagem no console de gerência, celular, etc.. E o segundo tipo que é a auditoria que consiste em periodicamente analisar os componentes críticos do sistema e procurar por mudanças suspeitas. Esse processo pode ser realizado por ferramentas que procuram, por exemplo, modificações no tamanho dos

arquivos de senhas, usuários inativos, etc. (NOBRE, 2007)

Finalmente o terceiro aspecto básico em um sistema de segurança refere-se a recuperação. Considera-se três tipos de recuperação de dados. A cópia de segurança dos dados (*backup*), que consiste em manter sempre atualizados e testados os arquivos de segurança em mídia confiável e separados física e logicamente dos servidores. O segundo tipo que são os aplicativos de *backup*, que são ferramentas que proporcionam a recuperação rápida e confiável dos dados atualizados em caso da perda das informações originais do sistema. E o terceiro tipo que a considerar que é o b*ackup* do h*ardware*, ou seja, a existência de hardware reserva, fornecimento autônomo de energia, linhas de dados redundantes, etc., que podem ser justificados levando-se em conta o custo da indisponibilidade dos sistemas. (NOBRE, 2007)

1.3 PRINCÍPIOS DE PREVENÇÃO E PROTEÇÃO

A análise das possíveis ameaças e riscos que a rede está submetida definem as formas de proteção da informação. Isso objetiva de manter sua confidencialidade, disponibilidade e integridade, e também para atender aos objetivos de gestão traçados pela alta direção. (NOBRE, 2007)

> As informações podem ser classificadas de acordo com o eventual impacto gerado decorrente de acesso, divulgação ou conhecimento não autorizado. Logo, a segurança de uma rede de computadores está relacionada à necessidade de proteção dessas informações contra acessos não autorizados ou contra a utilização indevida dos recursos computacionais, além de preservar a integridade dos dados armazenados contra a manipulação de qualquer natureza. (NOBRE, 2007,p.13)

Ainda de acordo com Nobre (2007) (a*pud* SILVA E TORRES, 2003, p. 17), a preservação da confidencialidade, integridade e disponibilidade da informação utilizada nos sistemas de informação requer medidas de segurança, que por vezes são também utilizadas como forma de garantir a autenticidade e o não repúdio.

As medidas de segurança podem ser classificadas em duas grandes categorias. Esta classificação se dá em função do modo como estas medidas abordam as ameaças. São elas, prevenção e proteção. A prevenção é o conjunto das medidas que visam reduzir a possibilidade das ameaças concretizarem-se. Essas medidas precisam ser implementadas antes da concretização da ameaça, ou

seja, antes do incidente ocorrer. O efeito das medidas de prevenção extingue-se quando uma ameaça se transforma num ataque. (NOBRE, 2007)

As medidas de proteção têm como objetivo dotar os sistemas de informação com ferramentas capazes de garantir a integridade da rede frente a um ataque, bloqueando acessos indevidos e, até mesmo, respondendo instantaneamente às tentativas de intrusão. (NOBRE, 2007)

1.4 SISTEMAS DE DETECÇÃO DE INTRUSÕES

Por analogia, compara-se um IDS a um alarme contra ladrões – um alarme contra ladrões configurado para monitorar locais de acesso, atividades suspeitas e invasores conhecidos. O modo mais simples de definir um IDS poderia ser descrevendo-o como uma ferramenta especializada que sabe como ler e interpretar o conteúdo dos arquivos de log de roteadores, firewalls, servidores e outros dispositivos da rede onde atua. Além disso, um IDS pode frequentemente armazenar um banco de dados com assinaturas de ataques conhecidas e assim, pode comparar padrões de atividade, tráfego ou comportamento que vê nos logs monitorados, com essas assinaturas, para reconhecer quando ocorre uma correspondência próxima entre uma assinatura e um comportamento corrente ou recente. (CASWELL, 2003)

Pode-se dizer que um IDS está para uma rede assim como um pacote de *software* antivírus está para os arquivos que entram em um sistema : ele inspeciona o conteúdo do tráfego da rede para procurar e desviar possíveis ataques, exatamente como um pacote de *software* antivírus inspeciona o conteúdo de arquivos, anexos de *e-mail*, conteúdo de *web* recebidos, etc., para procurar assinaturas de vírus (padrões que correspondem a *software* danoso conhecido) ou possíveis ações maldosas (padrões de comportamento que são no mínimo suspeitos, se não completamente inaceitáveis). (CASWELL, 2003)

Segundo Nobre (2007), os IDS são sistemas automáticos que funcionam como verdadeiros *sniffers* e, em tempo real, analisam o tráfego na rede e detectam tentativas não autorizadas de acesso à infra-estrutura lógica. O grande objetivo destes sistemas é proporcionar uma reação efetiva aos ataques que um segmento de rede possa vir a sofrer.

Desta forma, ainda de acordo com Nobre (2007), os IDS são frequentemente

considerados como uma das principais ferramentas defesa contra agressores, tornando-se um componente essencial para um bom sistema de segurança em rede. Eles atuam baseando-se nos tipos conhecidos de ataques e também verificando alterações de comportamento no tráfego de dados. Sempre que é detectada alguma alteração no comportamento desse tráfego ou identificado algum padrão de ataque, o sistema pode enviar um alerta aos administradores da rede, contra-atracar ou simplesmente se defender baseado em alguma configuração predefinida.

Um IDS não utiliza medidas preventivas, age como um informante assim que um ataque é detectado. *Firewalls* e *routes* bem configurados podem evitar a maioria dos ataques contra seus sistemas, porém, esta estrutura não tem visibilidade ao que acontece do lado de fora de seu perímetro. Além disso, o *firewall* não pode proteger o sistema de ataques realizados por meio de portas legítimas permitidas. Em contrapartida, o IDS pode, além de detectar ataques através das portas permitidas e oriundas do exterior, como também tem visibilidade do que acontece no perímetro da sua rede interna. (NOBRE, 2007)

1.4.1 Tipos de IDS

Os IDSs podem ser divididos em baseados em redes, baseados em *host* e distribuídos. Os IDSs que monitoram *backbones* de rede e procuram assinaturas de ataque são chamados *Network-Based Intrusion Detection System (*NIDS), enquanto aqueles que operam em *hosts* defendem e monitoram os sistemas operacionais e sistemas de arquivos contra sinais de invasão, são chamados *Host-Based Intrusion Detection System* (HIDS). (CASWELL, 2003)

1.4.1.1 NIDS

Os NIDS são assim chamados porque monitoram o tráfego da rede inteira. Mais precisamente, monitora um segmento da rede. Normalmente, uma *Network Interface Card* (NIC) de computador opera no modo não-promíscuo. Nesse modo de operação, apenas os pacotes destinados ao endereço *Mídia Access Control* (MAC) específico da NIC são encaminhados à pilha para análise. O NIDS deve operar no modo promíscuo para monitorar o tráfego da rede não destinado ao seu próprio endereço MAC. No modo promíscuo, o HIDS pode captar todas as comunicações do

segmento de rede. (CASWELL, 2003)

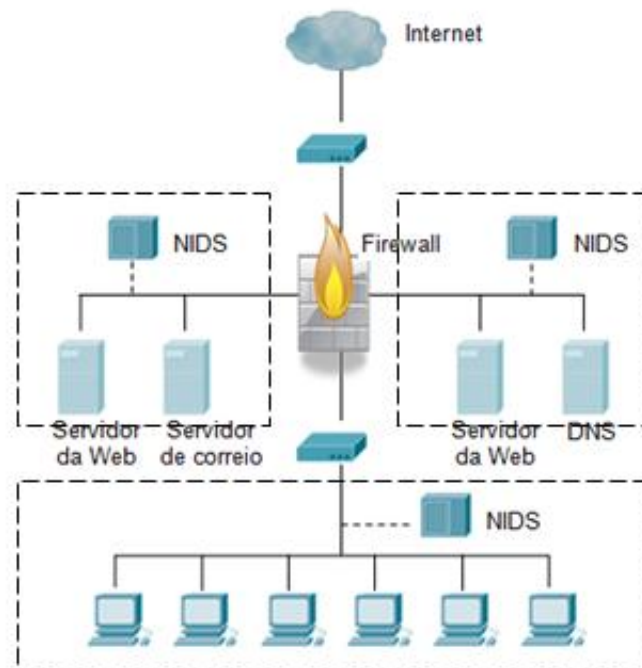



Figura 1.1 – Posicionamento dos NIDS e abrangência do segmento
Fonte : modificado pelo autor. Caswell, 2003, p.4.

Na Figura 1.1, observa-se uma rede usando três IDSs. As unidades foram colocadas em segmentos estratégicos da rede e podem monitorar o tráfego de todos os dispositivos que estão no segmento. Essa configuração representa uma topologia de rede de segurança de perímetro padrão, onde as subredes selecionadas que abrigam os servidores públicos são protegidas pelo IDS.

1.4.1.2 HIDS

O HIDS difere do NIDS de duas maneiras. O HIDS protege apenas o sistema *host* em que ele reside e sua placa de rede opera no modo não-promíscuo. Uma desvantagem do modo promíscuo é o de poder usar muito a CPU de uma máquina, tornando-a lenta.(CASWELL, 2003)

Uma vantagem do HIDS é a capacidade de personalizar o conjunto de regras para uma necessidade específica. Por exemplo, não há necessidade de examinar múltiplas regras projetadas para detectar explorações de *Domain Name System* (DNS) em um *host* que não está executando serviços de domínio.

Consequentemente, a redução do número de regras pertinentes melhora o desempenho e diminui a sobrecarga do processador. (CASWELL, 2003)

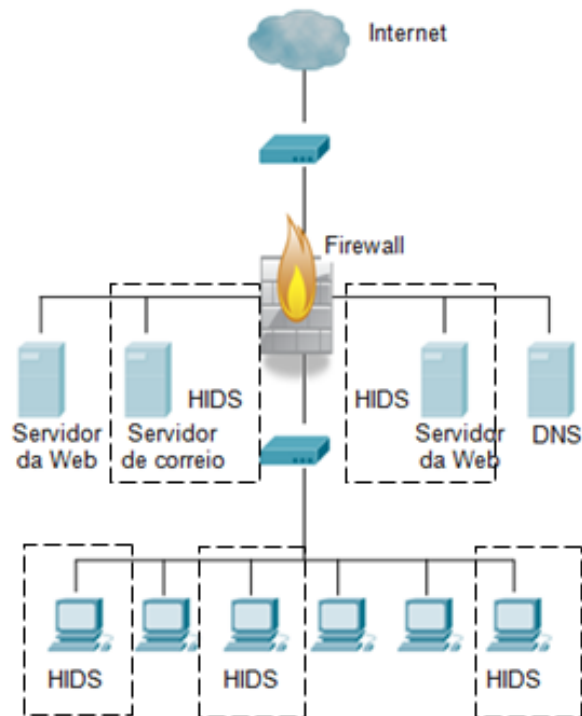



Figura 1.2 – Posicionamento dos HIDS e abrangência do *host*
Fonte : modificado pelo autor. Caswell, 2003, p.5.

1.4.1.3 Sistemas Híbridos

Segundo Nobre (2007), um sistema IDS híbrido une as vantagens dos dois tipos, HIDS e NIDS, a fim de proporcionar uma melhor capacidade de detecção de intrusões. O IDS híbrido funciona como o NIDS coletando o tráfego de pacotes da rede, processando as informações e detectando e respondendo a ataques do mesmo modo como ocorre no HIDS.

Com relação ao gerenciamento, alguns sistemas podem ter uma centralização dos IDS, pois alguns sensores, baseados em rede, são localizados em diversos segmentos de rede e outros IDS, baseados em host, são usados em servidores. O gerenciador pode controlar as regras dos dois tipos, formando o IDS híbrido. (NOBRE, 2007)

Ainda segundo Nobre (2007) (a*pud* STREBE E PERKINS, 2002, p. 288) sistemas IDS sempre requerem recursos da rede para funcionarem corretamente.

Sistemas NIDS usualmente funcionam em *firewalls* ou computadores dedicados; isso normalmente não é problema porque esses recursos são disponíveis. Entretanto, Sistemas HIDS destinados a proteger servidores podem ter sérias restrições para funcionamento. No caso dos servidores da zona neutra, a *DeMilitarized Zone* (DMZ) ou "zona desmilitarizada", o uso de IDS híbrido é vantajoso uma vez que ataques específicos a cada servidor podem ser identificados com maior precisão.

1.5 DEFICIÊNCIAS NOS IDS

Segundo Nobre (2007), outra questão potencialmente problemática tem a ver com o volume de dados gerado. Numa rede com elevados índices de atividade, os dados registrados pelos sensores podem atingir proporções significativas, o que implica dificuldades na capacidade de detecção e de gestão. Estes sistemas requerem acompanhamento em tempo real como forma de validação das ocorrências registadas, e se considera-se ainda que existem registros dos servidores, *routers*, *firewalls*, antivírus, etc., a carga administrativa associada pode ser muito grande.

> “Uma deficiência da maioria dos sistemas de detecção de intrusão é o fato do seu funcionamento se basear no mesmo princípio dos sistemas antivírus, ou seja, utilizam bases de dados com assinaturas de ataques: se ataques conhecidos são detectados, as tentativas são bloqueadas com relativa facilidade, caso contrário, podem passar impunemente.” (NOBRE, 2007,p.15)

A instalação de um IDS deve ser cuidadosamente avaliada, já que seu custo cresce proporcionalmente a capacidade de proteção desejada. No entanto, todo o esforço dedicado ao planejamento de instalação de um IDS poupará tempo na gestão da informação gerada. (NOBRE, 2007)

Ao lidar com os dados gerados, é necessário confirmar a sua aplicabilidade, ou seja, verificar se não se trata de falsos positivos: um falso positivo é a identificação de uma atividade legítima como sendo um ataque. Os responsáveis pela segurança da rede deverão analisar os dados obtidos e confirmar se são ataques reais ou não. (NOBRE, 2007)

Os dados relativos à atividade maliciosa registados pelo IDS podem ocasionar várias reações: alertas administrativos (incluindo chamadas para

celulares) e reações automáticas (interrupção da conexão ou bloqueio do IP de origem, por exemplo). (NOBRE, 2007)

1.6 METODOLOGIAS DE DETECÇÃO

De acordo com Lari e Amaral (2004), as formas de detecção são divididas sob dois aspectos: aqueles que analisam as informações baseadas em eventos passados (conhecimento) e aqueles que analisam o estado corrente do sistema (comportamento).

1.7 DETECÇÃO BASEADA EM CONHECIMENTO

Para Lari e Amaral (2004), os IDS são normalmente usados para auditar os dados gerados em busca de anomalias conhecidas e reportá-las ao administrador. Estes sistemas buscam por uma destas características: padrões de ataque conhecidos ou inconsistências que não podem ser geradas por operação normal. Além disso, eles tentam reduzir o grande volume de dados para proporcionar sumários indicativos de novas tendências.

Ainda segundo Lari e Amaral (2004), a grande vantagem desse tipo de abordagem é que, em teoria, produz pouco falso positivo. Ao possuir uma base de dados com assinaturas de ataques já conhecidos, é realizada uma checagem com o evento capturado pelo sistema. A partir daí são gerados alarmes que denunciam a presença de ações potencialmente danosas ao bom funcionamento. (LARI E AMARAL, 2004)

A necessidade de um conhecimento amplo do ambiente onde o IDS será instalado. Deve-se ter em mãos, para a elaboração das assinaturas, os sistemas operacionais aplicativos usados nos servidores e nas estações de trabalho, especificações de *hardware* e a topologia de sua rede interna, informação esta de grande valor, pois a quantidade de sensores a serem alocados na rede depende diretamente da topologia da mesma. (LARI E AMARAL, 2004)

Neste tipo de detecção, reduzir os dados gerados é o foco principal, visto que, o objetivo é eliminar informações que são irrelevantes à detecção, produzindo assim, relatórios reduzidos para investigação futuras. (LARI E AMARAL, 2004)

Uma contínua atualização das assinaturas é uma tarefa que requer uma

análise cuidadosa das vulnerabilidades que possam afetar o seu sistema como um todo, pois se não houver alguma assinatura que não tenha utilidade para seu ambiente, podemos elevar o número de falsos positivos. (LARI E AMARAL, 2004)

1.8 DETECÇÃO BASEADA EM COMPORTAMENTO

De acordo com Lari e Amaral (2004), a detecção baseada em comportamento assume que um evento pode ser detectado observando um desvio de comportamento esperado tanto dos sistemas como dos usuários. Este tipo de abordagem possui dois desafios básicos. O primeiro é caracterizar o que vem a ser comportamento normal e o segundo, é detectar desvios de maneira correta. Estes sistemas tendem a produzir muitos falsos positivos e resultados difíceis de serem interpretados. Os falsos positivos podem ser reduzidos, mas em contrapartida, pode-se elevar o número de falsos negativos.

Ainda de acordo com Lari e Amaral (2004), um ponto a ser destacado nesta abordagem é a possibilidade de detectar novos ataques e vulnerabilidades e de possuir uma dependência menor aos mecanismos do sistema operacional. A desvantagem reside no fato de que ataques que usam ações legítimas podem obter acesso ao sistema sem nos fornecer algum alarme.

1.9 POSICIONAMENTO DOS SENSORES

Para Nobre (2007), um dos problemas para a utilização de IDS, especificamente do NIDS, é a limitação de desempenho causada pela segmentação cada vez maior das redes pela utilização de *switches*, já que os NIDS trabalham em modo promíscuo analisando todos os pacotes que passam pelo segmento da rede. É possível utilizar o NIDS em redes segmentadas por *switches* usando sensores HIDS em conjunto com o NIDS, nos IDS híbridos. Os sensores podem ser usados de diversas formas, as quais irão refletir o grau de monitoramento do ambiente de rede.

1.9.1 Localização do IDS na Rede

De acordo com Nobre 2007, o IDS pode ser utilizado em diversas posições na rede e cada posição significa um tipo de proteção específico. Outra consideração

importante é quanto ao posicionamento do IDS em relação ao *firewall* da rede:

- Posicionado antes do *firewall*, a detecção é considerada simultânea aos ataques (detecção de ataques);
- Posicionado após o *firewall*, a detecção passa a ser de intrusões (detecção de intrusões) ou de erros cometidos pelos usuários internos (*misuse*).

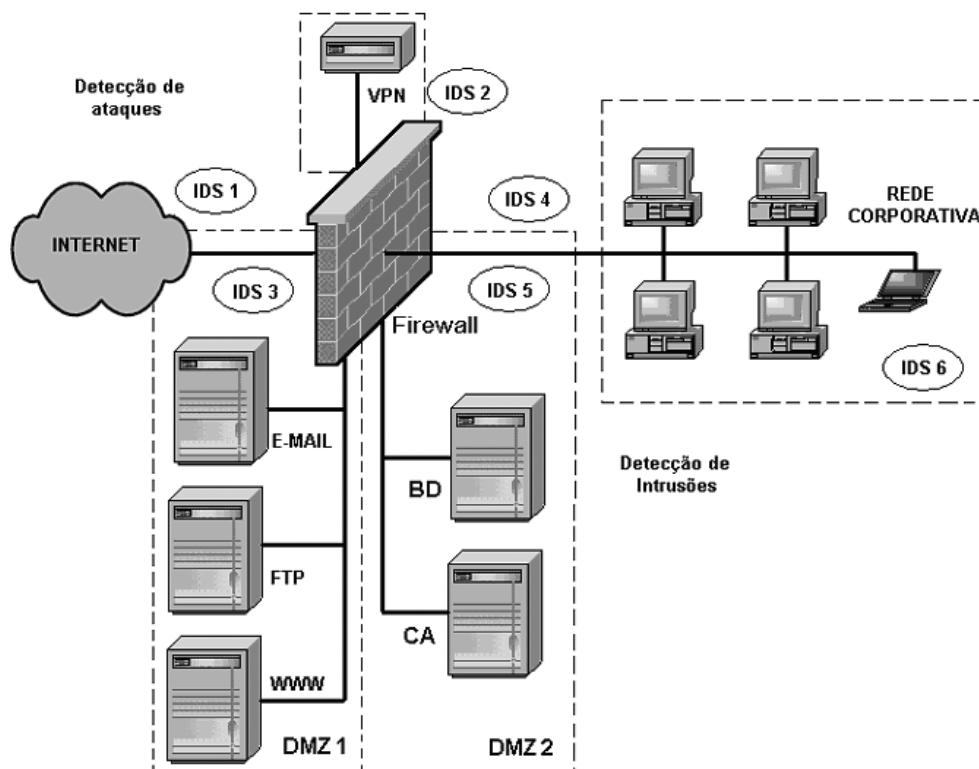



Figura 1.3 - Localização de IDS na rede
Fonte : Nobre, 2007, p.17

Na Fig. 1.3 observa-se os seguintes posicionamentos:

• **IDS 1 –** detecta tentativas de ataques externos, oferecendo uma fonte de informações sobre os tipos de ameaças de intrusão para a rede corporativa;

• **IDS 2 –** funciona no próprio *firewall*, detectando tentativas de ataque contra este;

• **IDS 3 –** detecta tentativas de ataque contra os servidores localizados na DMZ 1, que conseguem passar pelo *firewall*;

• **IDS 4 –** detecta tentativas de ataque contra recursos internos da rede que passaram pelo *firewall* e que podem ocorrer via VPN, por exemplo;

• **IDS 5 –** detecta tentativas de ataque contra os servidores localizados na DMZ 2, que passaram pelo *firewall*, pela VPN ou por algum outro serviço na DMZ 1;

• **IDS 6 –** detecta tentativas de ataques internos na rede corporativa.

1.10 SISTEMAS DE PREVENÇÃO DE INTRUSÕES

Segundo Nobre (2007), o funcionamento do IDS como *sniffer* apresenta alguns problemas, como o fluxo de pacotes fragmentados, não confiáveis e que chegam fora de ordem. Os sistemas funcionam em modo passivo, apenas escutando o tráfego, não sendo capazes de controlar esse tráfego, seja ignorando, modificando, atrasando ou injetando pacotes para defender a rede.

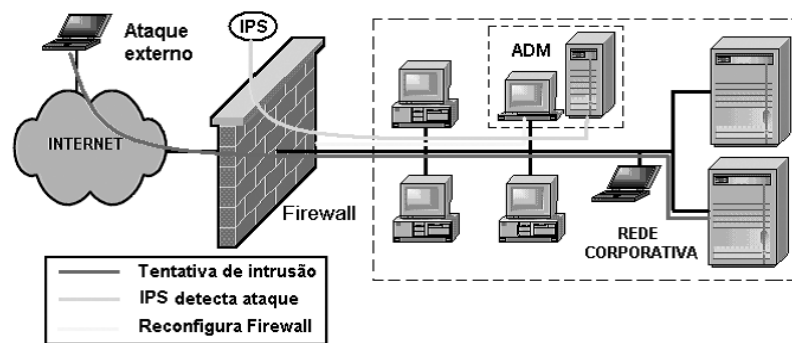



Figura 1.4 - Funcionamento do IPS na rede
Fonte : Nobre, 2007, p.18

Ainda para Nobre (2007), a identificação desses pontos fracos, relacionados a determinados tipos de IDS, levou ao desenvolvimento de novos sistemas que buscam detectar e prevenir ataques contra a rede de comunicação. Operando de forma *inline* na prevenção de intrusões, eles são conhecidos como Sistemas de Prevenção de Intrusões *Intrusion Prevention System* (IPS).

A operação *inline* difere da operação passiva na forma de capturar os pacotes dos segmentos de rede. Enquanto o IDS que opera no modo passivo captura o tráfego do segmento de rede, o IDS que opera no modo *inline* assume uma posição como um *firewall*, onde todo o tráfego da rede passa por ele, como o que é mostrado na Figura 1.4. Essa característica permite que o IDS *inline* seja capaz de detectar os ataques e, além disso, também de preveni-los através de bloqueio, pois os pacotes dos atacantes não chegam aos servidores da rede. (NOBRE, 2007)

1.11 DETECÇÃO DE ANOMALIAS

Para Nobre (2007), uma anomalia é definida como algo diferente, anormal, peculiar ou que não seja facilmente classificado. No contexto de segurança de computadores, uma anomalia pode ser definida como ações ou dados que não sejam considerados normais por um determinado sistema, usuário ou rede.

Ainda para Nobre (2007), essa definição abrange ainda uma grande variedade de itens e pode incluir tópicos como padrões de tráfego, atividades dos usuários e comportamento de aplicativos. Acredita-se que pelo menos uma significativa porção das ameaças ou condições que causem alarme deve manifestar-se como anomalias, sendo assim detectáveis. A maioria dos sistemas de detecção de anomalias que se concentram em segurança normalmente se enquadra em três categorias gerais: comportamental, padrão de tráfego ou protocolo.

1.11.1 Anomalias em Padrões de Comportamento

Segundo Nobre (2007), sistemas de detecção que procuram por anomalias em padrões de comportamento (normalmente o comportamento de usuários), são normalmente de características, porém eles podem abranger também alguns critérios de estatísticas, como os tipos de aplicativos e protocolos usados em várias horas do dia, a relação entre a origem e o destino das atividades da rede ou até mesmo os tipos de anexos de e-mail que são enviados através de um sistema.

1.11.2 Anomalias em Padrões de Tráfego

De acordo com Nobre (2007), os sistemas de detecção que procuram por anomalias em padrões de tráfego da rede, normalmente são de natureza estatística, apesar de incluírem algumas características como volume de tráfego, mistura de protocolos e várias distribuições na origem e no destino.

Ainda de acordo com Nobre (2007), para ilustrar, pode-se considerar o gerenciamento de uma rede ou sistemas simples de monitoração de negação de serviços, que possuem a vantagem de operar em um domínio muito maior e variado, e que podem ser criados a partir de um número de bons modelos de estatísticas. A desvantagem é que esses sistemas freqüentemente não são capazes de detectar a

maioria das anomalias qualitativas ou quantitativas sutis. Eles apresentam também algumas dificuldades na definição de uma base confiável para o desempenho da análise de estatísticas.

1.11.3 Anomalias em Padrões de Protocolos

Segundo Nobre (2007), os sistemas de detecção que procuram por anomalias em padrões de protocolos são normalmente sistemas de características. Estes tendem a variar um pouco de acordo com a implementação, mas os mais eficientes são freqüentemente implementados como sistemas de modelo rígido. Esse tipo de sistema tira proveito do fato de que os protocolos sozinhos são geralmente muito restritos. Eles tendem a limitar muito a natureza e ordem das transações e são geralmente muito bem descritos por alguma implementação ou documento de referência.

Ainda de acordo com Nobre (2007), outra vantagem desse sistema é que ele pode detectar uma grande variedade de anomalias dentro do espaço do protocolo, podendo ser construído com muita eficiência. A desvantagem, porém, é que pode ser difícil de estimar o efeito da anomalia observada de forma acurada, uma vez que alguns tipos de transações de protocolo problemáticas (como ataques, por exemplo) não se manifestam como anomalias.

1.12 TESTES DE INTRUSÃO

Os testes de intrusão são tentativas de acesso aos sistemas da rede corporativa por parte de pessoas autorizadas, que podem ser realizados sem qualquer conhecimento prévio dos sistemas a testar, ou com a indicação das respectivas características. Estes testes podem ser realizados tanto a partir do interior da rede como a partir da Internet, dependendo a decisão sobre o ponto inicial do teste, das características da organização (número de sistemas, número de funcionários, tipo de aplicações) e dos resultados que possam ser obtidos da análise. (NOBRE, 2007)

1.13 SEGURANÇA INTEGRADA

De acordo com Nobre (2007), esse método combina várias tecnologias de

segurança com compatibilidade de políticas, gerenciamento, serviço e suporte, e pesquisa avançada para uma proteção mais efetiva. Através da combinação de várias funções, a segurança integrada pode proteger a rede com mais eficiência contra a variedade de ameaças e, em cada nível, para minimizar os efeitos dos ataques.

As tecnologias de segurança principais que podem ser integradas incluem:

- ***Firewall* -** Controla todo o tráfego de dados através da verificação das informações que entram e saem da rede a fim de garantir que não ocorram acessos não autorizados;
- **Detecção de Intrusão -** Detecta o acesso não autorizado e fornece diferentes alertas e relatórios que podem ser analisados para políticas e planejamento da segurança;
- **Filtragem de Conteúdo -** Identifica e elimina o tráfego de pacotes não desejado na rede;
- **Redes Privadas Virtuais (VPN) -** Asseguram as conexões além do perímetro da rede local, permitindo que redes locais se comuniquem com segurança através da Internet;
- **Gerenciamento de Vulnerabilidade -** Permite a avaliação da posição de segurança da rede descobrindo falhas de segurança e sugerindo melhorias;
- **Proteção Antivírus -** Protege contra vírus, *worms*, Cavalos de Tróia e outras pragas virtuais.

Ainda de acordo com Nobre 2007, Individualmente, essas tecnologias de segurança podem ser incômodas para instalar e geralmente são difíceis e caras de gerenciar e atualizar. Entretanto, quando integradas em uma solução única, elas oferecem uma proteção mais completa enquanto a complexidade e o custo de operação e manutenção são reduzidos.

1.14 CONSIDERAÇÕES FINAIS

Este capítulo mostra que o cuidado com a segurança de redes deve ser observada com muito critério. É um assunto complexo que envolve inúmeros conceitos e a criação de políticas de segurança é imprescindível.

Há a necessidade de se desenvolver uma visão ampla dos sistemas de

detecção de intrusão, para uma melhor de configuração, considerando-se o ambiente ao qual ele será implementado.

O próximo capítulo fala sobre o IDS de distribuição gratuita chamado *Snort*, um dos sistemas de detecção de intrusão mais utilizados para sistemas Linux.

2 *SNORT*

O objetivo deste capítulo é mostrar a estrutura do *Snort*. Sua origem, como funciona seu mecanismo de detecção e a análise dos pacotes, o pré-processador, e o uso das regras do *Snort* que funcionam como assinaturas pré-registradas de tentativas de invasão conhecidas.

2.1 DEFININDO O *SNORT*

De acordo com Caswell (2003), o *Snort* é um sistema de detecção de invasão de rede de código-fonte aberto, capaz de realizar análise de tráfego em tempo real, em redes *Internet Protocol* (IP). O *Snort* pode realizar analise de protocolo, pesquisa/correspondência de conteúdo, ou seja, capacidade de inspecionar o *payload* do pacote, e ser usado para detectar uma variedade de ataques.

"Desde 1998, quando o *Snort* teve a primeira versão divulgada, percebemos o preenchimento de uma lacuna nos mecanismos de segurança." (LARI e AMARAL, 2007, p.24)

Segundo Lari e Amaral (2007), o *Snort* fornece aos administradores de rede informações importantes na tomada de decisões mediante atividades suspeitas que ocorram na rede, pois o mesmo faz uma análise detalhada do tráfego em questão. Uma característica importante do *Snort* está na sua capacidade de inspecionar o *payload* do pacote, ou seja, a área onde encontra-se o conteúdo do pacote.

2.2 ARQUITETURA

De acordo com Lari e Amaral (2007), o *Snort* é considerado um IDS "leve", ou seja, pode ser colocado em funcionamento com muito pouco esforço tanto no sentido computacional quanto no sentido de configuração e suporte. Sendo assim, seus recursos podem ser melhor aproveitados com a checagem de pacotes e não com IDS em si. A base do *Snort* está montada sobre a biblioteca LibPcap que provê funções de acesso a recursos de rede de baixo nível, como monitoramento de segurança.

Conforme Caswell (2003), existem três modos principais nos quais os o

Snort pode ser configurado, sendo estes : farejador (*sniffer*), registrador de pacotes (*packet logging*) e detecção de invasão. O modo farejador simplesmente lê os pacotes que trafegam na rede e os exibe continuamente no console. O modo registrador de pacotes faz esta leitura do tráfego e grava os pacotes em disco. Já o modo de detecção de invasão é mais complexo, pois faz com que o *Snort* analise os pacotes da rede procurando correspondências em relação a um conjunto de regras pré-definidas (assinaturas), executando diversas ações com base no que encontra.

Ainda conforme Caswell (2003), o *Snort* possui vários componentes importantes, a maioria deles exige *plug-ins* para personalizar sua implementação. Esses componentes incluem pré-processadores e *plug-ins* de alerta, que permitem ao *Snort* manipular um pacote para tornar o conteúdo mais administrável pelo mecanismo de detecção, e o sistema de alerta, que pode enviar sua saída através de vários métodos. O *Snort* consiste em quatro componentes básicos : farejador (sniffer), pré-processador, mecanismo de detecção e saída.

Segundo Caswell (2003), em sua forma mais básica, o *Snort* é um farejador de pacotes. Porém, ele é projetado para processar os pacotes através do pré-processador e depois procurar uma correspondência desses pacotes com uma série de regras (através do motor de detecção). Como ilustra a Figura 2.1.

Figura 2.1 - Arquitetura do Snort
Fonte : Caswell, 2003, p.26. Modificado pelo autor.

O farejador de pacotes é um dispositivo (de *hardware* ou *software*) usado para fazer escuta em redes. Ele funciona de maneira semelhante a uma escuta telefônica, mas é usado para redes de dados, em vez de redes de voz. (CASWELL, 2003)

Conforme Caswell, (2003), um farejador de rede permite que um dispositivo de *software* ou *hardware, "escute"* o tráfego da rede de dados. No caso da Internet, isso normalmente consiste em tráfego IP, mas pode ser outro tipo de tráfego. Como

o tráfego IP consiste em tipos diferentes de tráfego de rede, como, *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Internet Control Message Protocol* (ICMP), protocolos de roteamento, etc, muitos farejadores analisam os vários protocolos de rede para transformar os pacotes em algo legível para seres humanos.

Os farejadores de pacotes tem vários usos: análise, diagnóstico e solução de problemas de rede, análise e comparativo de desempenho e obter senhas em texto puro e outros dados interessantes.

2.3 PRE-PROCESSADOR E MECANISMO DE DETECÇÃO

De acordo com Caswell (2003), o pré-processador captura os pacotes brutos e os verifica em relação a certos *plug-ins*. Esses *plug-ins* conferem certo tipo de comportamento do pacote. Uma vez determinado que o pacote tem um tipo em particular de “comportamento”, ele é então enviado para o mecanismo de detecção, como mostra a figura 2.2.

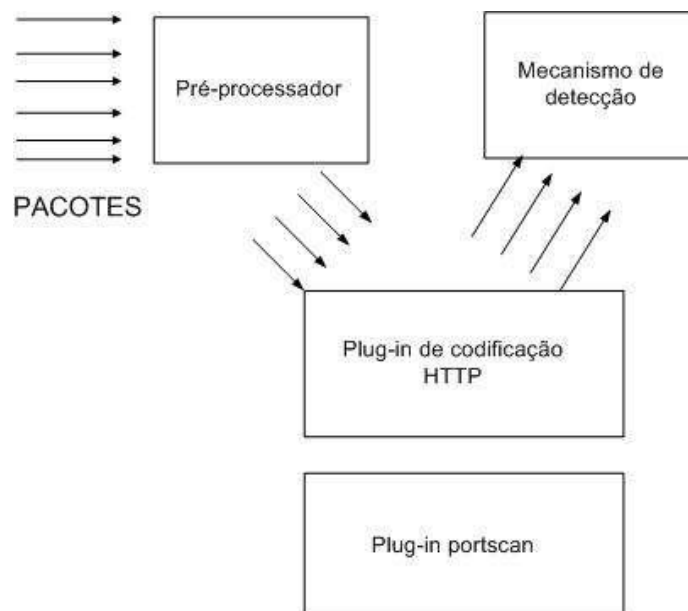



Figura 2.2 - Pré-processador do Snort (CASWELL, 2003)
Fonte : Caswell, 2003, p.28.

Ainda de acordo com Caswell (2003), o mecanismo de detecção ou motor de detecção, é a parte mais importante do IDS no *Snort*. Ele captura os dados provenientes do pré-processador e seus *plug-ins*, e esses dados são comparados com um conjunto de regras. Se há correspondência dos dados do pacote com

alguma das regras, então eles são enviados para o processo de registro/alerta. Os conjuntos de regras são agrupados por categoria (cavalos de Tróia, estouro de *buffer*, acesso a vários aplicativos) e são atualizados regularmente.

As regras consistem em duas partes:

- O cabeçalho da regra: é basicamente a ação de executar (*log* ou alerta), o tipo de pacote de rede (TCP, UDP, ICMP etc.), endereços IP e portas de origem e destino;
- A opção da regra: é o conteúdo do pacote que deve fazer com que ele corresponda à regra.

Segundo Lari e Amaral (2004), o *Snort* armazena suas regras (assinaturas) de uma maneira bastante simples mas, ao mesmo tempo, poderosa. Consiste em uma lista ligada bidimensional, com dois campos denominados *Chain Headers* e *Chain Options*. Os atributos comuns às regras (endereços IP de origem e destino e portas de origem e destino) são armazenados no *Chain Headers*. Todos os outros atributos disponíveis em uma regra ficam no *Chain Options*. Assim, ao avaliar um pacote de acordo com uma de suas assinaturas, o *Snort* procura no campo *Chain Headers* da lista as características básicas da regra.

Ao encontrar a Chain correspondente a regra, o *Snort* procura os atributos restantes no campo *Chain Options*. Esta estrutura acaba tomando um formato como mostra a Figura 2.3.

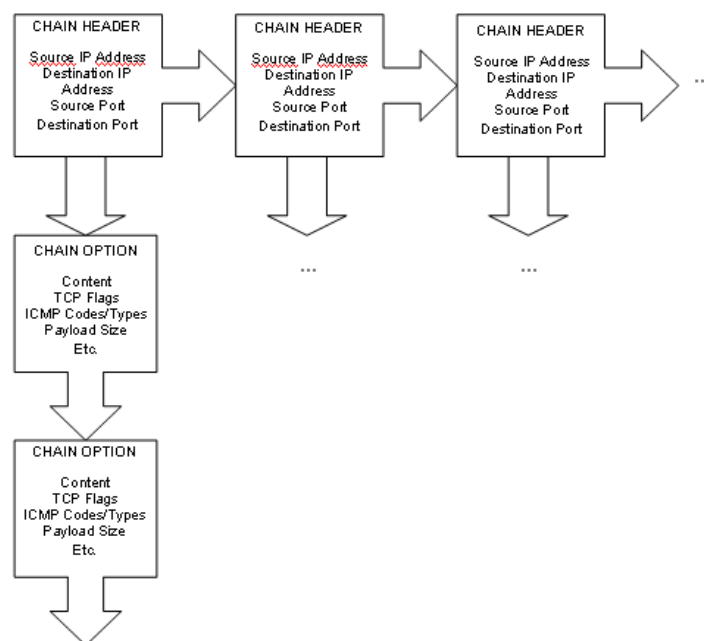



Figura 2.3 – Verificação de regras do *Snort*.
Fonte : Lari e Amaral, 2004, p.27.

Na curvatura de aprendizado do *Snort*, a mais íngreme refere-se a entender o mecanismo de detecção e suas regras. O *Snort* tem uma sintaxe específica que ele usa com suas regras. A sintaxe da regra pode envolver o tipo de protocolo, o conteúdo, o comprimento, o cabeçalho e vários outros elementos, incluindo caracteres de lixo para definir regras de estouro de *buffer*. A Figura 2.4 mostra o funcionamento do mecanismo de detecção. (CASWELL, 2003)

Após passarem mecanismo de detecção do *Snort*, os dados precisam ir para algum lugar. Se estes dados tem correspondência com alguma regra no mecanismo de detecção, então um alerta é disparado. Os alertas podem ser enviados para um arquivo de *log*, através de uma conexão de rede, através de *sockets* UNIX ou Windows Popup (SMB), ou de interrupções *Simple Network Management Protocol* (SNMP). Os alertas também podem ser armazenados em um banco de dados SQL, como *MySQL* e *Postgres*. (CASWELL, 2003)

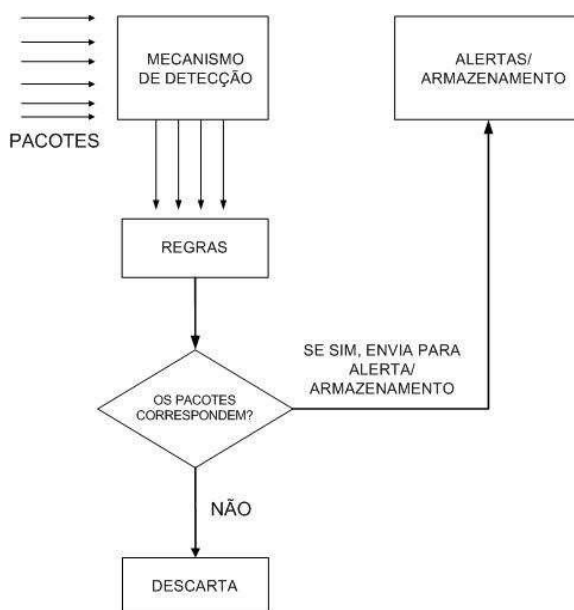



Figura 2.4 - Mecanismo de detecção do Snort.
Fonte : Caswell, 2003, p.29.

2.4 SISTEMA DE ALERTA E LOG

Conforme Lari e Amaral (2004), o sistema *log* do *Snort* é selecionado em tempo de execução através de linhas de comando. As opções são : logar pacotes na forma normal, codificada; logar no formato mais inteligível ou logar no formato TCPDump. A escolha da opção depende exclusivamente da performance desejada dos sistemas. O formato do TCPDump é o mais econômico nesse sentido, pois

condensa as informações em poucos campos. Também é possível desligar os logs completamente, melhorando ainda mais a performance.

Ainda conforme Lari e Amaral (2004), os alertas, por outro lado, podem ser enviados para o *syslog* do UNIX, para arquivos texto em dois formatos diferentes ou para mensagens *WinPopup*, usando o cliente Samba num sistema Windows.

Assim como o mecanismo de detecção e o pré-processador, o componente de alerta usa *plug-ins* para enviar os alertas para banco de dados e através de protocolos de interligação em rede, como *traps* SNMP e mensagens *WinPopup*. A Figura 2.5 ilustra esse funcionamento. (CASWELL, 2003)

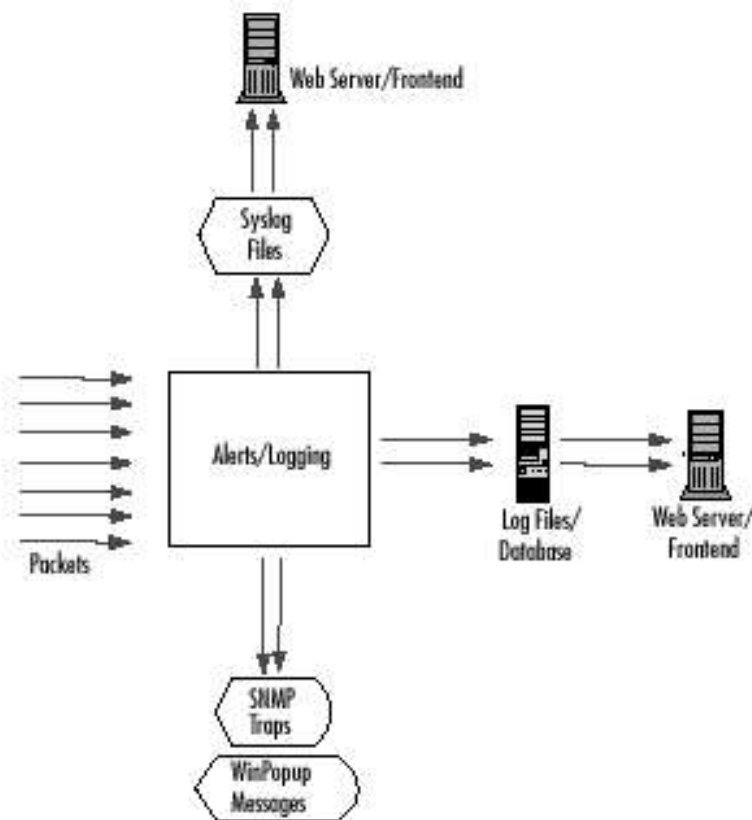



Figura 2.5 - Componente de alerta do *Snort*.
Fonte : Caswell, 2003, p.30.

2.5 FUNCIONAMENTO DO *SNORT*

Conforme Lari e Amaral (2004), para o *Snort* funcionar, é necessário instalar a libpcap, pcap (*packet capture*), uma biblioteca para capturar pacotes na rede. O Snort funciona de três maneiras diferentes :

Sniffer : com placa de rede funcionando em modo promíscuo, a mesma

captura pacotes e mostra os mesmos no formato que desejar. Abaixo estão alguns comandos básicos para funcionamento em modo *sniffer*. (LARI E AMARAL, 2004)

Para capturar os pacotes e imprimir os cabeçalhos do pacote TCP/IP :

```
./ snort -v
```

Se desejar analisar o cabeçalho IP e TCP/UDP/ICMP e os dados existentes na camada de aplicação : (LARI E AMARAL, 2004)

```
./ snort -vd
```

Para mostrar os cabeçalhos IP e TCP/UDP/ICMP, os dados do pacote e o cabeçalho da camada de enlace : (LARI E AMARAL, 2004)

```
./  snort -vde
```

Registrador de pacotes : grava os pacotes capturados no disco rígido. Os comandos acima mostrados são bastante úteis, mas se deseja registrar os pacotes capturados é necessário especificar um diretório, no caso /var/log/snort, então faríamos : (LARI E AMARAL, 2004)

```
./ snort –vde –l /var/log/snort
```

Quando se usa o parametro –l, o Snort não possui um critério para armazenar os registros em diretórios, usando o endereço IP do computador remoto ou endereço IP de sua rede interna como nome de diretório. Assumindo que a rede interna possui o endereço IP 192.168.1.0-192.168.1.255, para organizar em diretórios que tenham os endereços IP de computadores remotos como nomes dos subdiretórios, acrescenta-se uma opção na linha de comando anterior : (LARI E AMARAL, 2004)

```
./ snort –vde –l /var/log/snort –h 192.168.1.0/24
```

Uma vez que os pacotes foram registrados em um arquivo binário, pode-se ler os pacotes com qualquer sniffer que aceite o formato binário do tcpdump, como o próprio tcpdump ou o ethereal. O Snort também pode ler os pacotes usando o parâmetro –r, que o coloca na modalidade playback. Utiliza-se o seguinte comando : (LARI E AMARAL, 2004)

```
./ snort –dv –r  /var/log/snort/<packet.log>
```

Detecção de intrusão, isto é, analisa o tráfego de rede e compara com um conjunto de regras pré-configuradas (assinaturas) e exibe alertas ou toma alguma atitude caso haja correspondência de entre o conteúdo dos pacotes e algumas destas regras. Lari e Amaral (2004) afirmam que para habilitar o Snort no modo de detecção de intrusão, mostrando pacotes no console, usamos o parâmetro que

indica o arquivo *snort.conf,* onde estão localizadas as regras, como mostra o comando :

```
./ snort –vde –c /etc/snort/snort.conf
```

2.5.1 Filtros e criação de assinaturas

Segundo Lari e Amaral 2004, uma vez instalado, o IDS deverá fornecer somente o tráfego que interessa, deixando de lado o tráfego da rede que é considerado normal. Para ajudar nesta tarefa, será preciso "avisar" aos sensores o que filtrar. A assinatura define ou descreve um tráfego de rede nos moldes do que interessa, e a tarefa do responsável é aprender a observá-la e entendê-la. O filtro transcreve a descrição da assinatura para um código que é possível ser lido pelo computador no qual está instalado o IDS.

2.5.1.1 Políticas de filtragem

Conforme Lari e Amaral (2004), muitos ataques são impedidos de terem êxito devido à política de segurança implementada no *firewall*. A política do *firewall* ou de filtragem do roteador é governada por duas escolhas : negar tudo (*deny everything*) ou permitir tudo (*allow everything*). A política de filtragem de pacotes do *firewall* ou do roteador é o melhor companheiro do IDS.

2.5.1.1.1 *Deny Everything* (Negar tudo)

De acordo com Lari e Amaral (2004), "negar tudo" implica em muito mais que políticas de segurança. Esta regra nega o que não for especificamente permitido, isto é, o sensor deverá possuir uma regra para cada tráfego que deseja permitir sua entrada no perímetro de defesa. Se o tráfego não conseguir achar uma regra para esta situação, terá o pacote descartado. Esta regra facilita muito o desempenho do IDS, pois é só configurar uma regra para violação das políticas do *firewall*. Este é um ponto importante, pois a política de "negar tudo" pode deixar a sua estação sem comunicação com as outras estações, o que pode-se caracterizar como um famoso ataque *Denial of Service* (DoS), feito pelo administrador do próprio sistema, sem a necessidade de incursões de *hackers*, *script-kiddies* ou outras

atividades maliciosas.

2.5.1.1.2 Allow Everything (permitir tudo)

Ainda de acordo com Lari e Amaral (2004), “permitir tudo não especificamente negado” é uma política de *firewall* de alto risco, mas pode ser necessário em situações em que a liberdade e flexibilidade são valores que estão acima da segurança, como em laboratório de pesquisa ou em instituições educacionais. Esta abordagem faz com que a implementação e sua posterior customização seja trabalhosa, pois o IDS ficará encarregado de alarmar somente os eventos que violaram as políticas implementadas. Com esta política fica muito difícil detectar eventos maliciosos que podem causar danos ao ambiente da rede. Este princípio fere a teoria do IDS – manter relação com as políticas de filtragem do roteador e do *firewall*.

2.5.2 Assinaturas

Conforme Lari e Amaral (2004), uma das premissas no assunto que envolve a segurança de redes é a segurança integrada. Um *firewall* com suas políticas implementadas deve sempre estar relacionado com IDS e assim fornecer uma solução única para o perímetro de defesa da rede. Infelizmente, os analistas evitam agregar as políticas de filtragem do *firewall* e apenas incorporam os filtros de ataques já bem conhecidos pela comunidade. Este fato acontece decorrente de três situações :

- O analista não tem conhecimento sólido de TCP/IP;
- As equipes de *firewall* e detecção de intrusão não se comunicam e não compartilham informações;
- O analista não possui tempo para otimizar a solução de segurança que possui, é um caso típico de pequenas empresas em que o analista é responsável por tudo em seu ambiente (bancos de dados, *proxy*, IDS, firewall, correio eletrônico e aplicativos pertinentes ao ambiente).

Com o resultado deste desencontro de informações, poderemos ter um IDS sobrecarregado, com o desempenho afetado pela grande quantidade de informações que o mesmo deverá checar. (LARI E AMARAL, 2004)

2.5.2.1 Composição das assinaturas

As assinaturas do *Snort* são divididas em duas partes lógicas; o cabeçalho e opções. O cabeçalho contém a ação da assinatura, protocolo, endereços IP de origem e de destino e sub-máscara de rede, informações sobre portas de destino e origem. A parte de opções contém mensagens de alertas e informações sobre que parte do pacote deverá ser examinada para determinar que ação deverá ser tomada mediante o evento ocorrido. (LARI E AMARAL, 2004)

2.5.2.2 Ações

O cabeçalho contém informações sobre “quem, onde e o que” de um pacote, como também o que deve ser feito em caso de algum pacote se confrontar positivamente com uma das regras do *Snort*. (LARI E AMARAL, 2004)

Existem 5 ações a serem tomadas : *alert, log, pass, activate e dynamic*.

- *Alert* : gera um alerta usando o método selecionado e depois grava o pacote.
- *Log* : registra o pacote.
- *Pass* : ignora o pacote.
- *Activate* : alerta e ativa uma outra regra dinâmica.
- *Dynamic* : permanece ocupado até ser ativado por uma regra ativa e depois age como uma regra para registro.

Para ilustrar, o exemplo a seguir criará um tipo de regra que registrará os eventos no *syslog* (utilitário do UNIX) e no bando de dados MySQL. (LARI E AMARAL, 2004)

```
ruletype redalert { Type alert output
Alert_systlog : LOG_AUTH LOG_ALERT output database: log,
mysql, user=snort
dbname=snort host= localhost }
```

2.5.2.3 Protocolos

Neste campo da assinatura está o protocolo. O *Snort*, um excelente sistema de detecção de intrusão capaz de analisar vários protocolos com tráfego suspeito

com : TCP, UDP, ICMP e IP. Atualizou o tratamento dos protocolos HTTP e DCE / RPC , SIP, POP e IMAP3. Atualizações para o pré-processamento de HTTP e DCE / RPC agora permitem o Snort remontar as solicitações e respostas, mesmo quando distribuídos por pacotes de muitos, e de forma inteligente limpar os resultados. O *Snort* realiza a análise em tempo real sobre o tráfego de rede IP para detectar tentativas de sonda ou atacar a rede usando um conjunto de regras definidas pelo usuário que caracteriza esses ataques.

O pré-processador SIP melhorado pode identificar os canais de chamada e detectar anomalias em comunicações SIP. Os pré-processadores POP3 e IMAP são capazes de decodificar anexos de *email* em Base64 e uuencoded e o pré-processador SMTP é agora capaz de lidar com os dois últimos formatos. Um pré-processador de reputação experimental IP permite o *Snort* criar lista uma negra ou lista branca pacotes com base no endereço IP.

Outras melhorias incluem suporte para leitura de arquivos grandes e registro pcap URLs HTTP, nomes de arquivos anexos e os destinatários de e-mail ao gerar eventos. Há também atualização das regras e opções, melhor portabilidade para desenvolvedores e documentação melhorada.

Apesar de poder decodificar outros tipos de protocolos, o *Snort* é focado na pilha de protocolos TCP/IP. Usando a pilha de protocolos TCP/IP, a Figura 2.6 ilustra o nível de atuação dos componentes do *Snort*. (LARI E AMARAL, 2004)

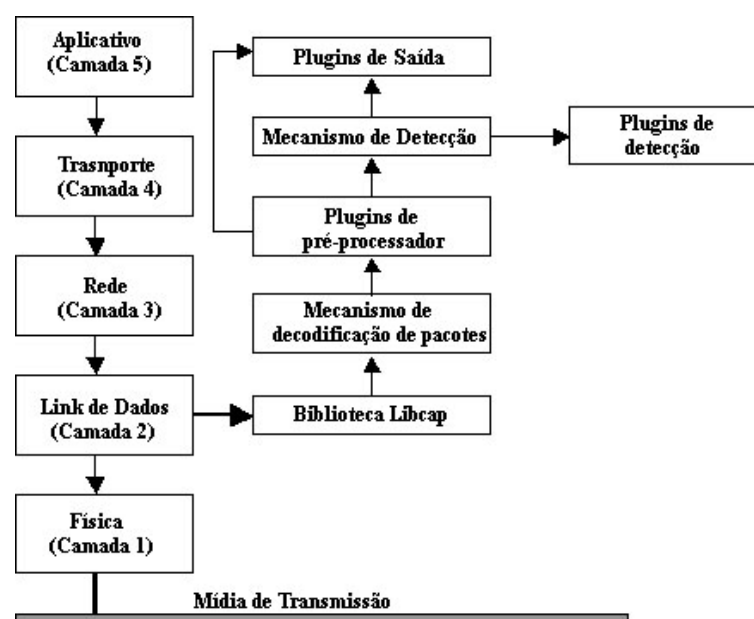



Figura 2.6 – Camadas TCP/IP.
Fonte : Caswell, 2003, p.73.

As camadas do TCP/IP :

- Camada de aplicação (5) – Por exemplo, protocolo *HiperText Transfer Protocol* (HTTP*)*, *Simple Mail Transfer Protocol* (SMTP*);*
- Camada de transporte (4) – Por exemplo, protocolo *TCP* e *UDP;*
- Camada de rede (3) – Por exemplo, protocolo *IP* e *ICMP;*
- Camada de enlace (2) – Por exemplo, protocolo *Ethernet*, *Token Ring;*
- Camada física (1) – Por exemplo, placa de rede, cabos, conectores.

2.5.2.4 Endereços IP

De acordo com Lari e Amaral (2004), no neste campo do cabeçalho da regra estão os endereços IP e as portas de conexão. A palavra "*any*" pode ser usada para definir qualquer endereço. Os endereços são representados da seguinte forma : o número IP seguido do CIDR (*Classless Inter-Domain Routing*). O bloco CIDR de máscara /24 indica uma rede classe C, /16 uma rede classe B e /32 especifica um endereço exato de uma máquina. Por exemplo, uma combinação mostra a configuração 192.168.1.0/24 e teria como alvo todas as estações com endereços de 192.168.1.1 a 192.168.1.255. No exemplo a seguir, é descrita uma regra que alerta qualquer conexão originada de sua rede interna em direção a seus endereços privativos.

```
Alert tcp ![192.168.1.0/24,10.1.1.0/24] any->
[192.168.1.0/24,10.1.1.0/24] 111 (contente: "|00 01 86 a5|";
msg: "Acesso externo ";)
```

2.5.2.5 Número de portas

O número de porta pode ser especificado de várias maneiras, incluindo "*any*", portas estáticas, uma faixa de números e pela negação. (LARI E AMARAL, 2004)

Exemplo : `log udp any any -> 192.168.1.0/24 1:1024 log udp`

Tráfego vindo de qualquer porta e com destino de portas variando de 1 a 1024.

2.5.2.6 Operador de direção

Segundo Lari e Amaral (2004), o operador de direção indica que sentido o pacote irá percorrer e assim a regra irá atuar de forma a tomar a medida correta. O endereço IP e número da porta que ficam no lado esquerdo indicam a origem da conexão, e o endereço IP e número da porta que ficam no lado direito indicam o destino da conexão.

Exemplo : `log tcp any any -> 192.168.1.0/24 23`

Esta regra registra todos os pacotes que tenham origem em qualquer endereço IP e qualquer porta, e como destino, a faixa de endereços de 192.168.1.1 a 192.168.1.255 na porta 23. (LARI E AMARAL, 2004)

2.5.2.7 Opções de regra

De acordo com Lari e Amaral (2004), as opções que as regras do Snort possue são o ponto chave do detector de intrusão, combinando flexibilidade e facilidade de uso. Todas as opções são separadas por ponto-e-vírgula (;). Seguem algumas palavras-chaves disponíveis :

msg : imprime uma mensagem em alerta e registra o pacote

logto : registra o pacote em um arquivo específico no lugar de um arquivo padrão.

ttl : testa o valor do campo TTL do cabeçalho IP

tos : testa o valor do campo TOS do cabeçalho IP

id : testa o campo de identidade da fragmentação do cabeçalho IP

ipoption : observa os campos de opções do IP

dsize : testa o tamanho do payload do pacote contra um valor fornecido

flags : testa os “*flags*” do TCP

content : procura por um molde no conteúdo do pacote

depth : configura um tamanho máximo no qual a procura deverá ser feita

resp : ativa uma resposta para um dado comportamento do pacote

reference : referência para ataques externos

sid : identidade da regra so Snort

rev : número da revisão da regra

classtype : identificador da classificação

priority : identificador do grau de impacto da regra

ip_proto : valor do protocolo no cabeçalho do IP

sameip : determina se o endereço IP de origem é igual ao endereço de destino

window : testa o tamanho do pacote TCP para um valor especificado

react : ativa uma resposta. Ex. bloquear algum site

seq : testa o número de *sequence* do TCP contra um valor fornecido

ack : testa o número de *acknowledgement* do TCP contra um valor fornecido.

2.6 CONSIDERAÇÕES FINAIS

Este capítulo objetivou mostrar o *Snort* na sua forma conceitual, assim como, a maneira como funcionam os aspectos de análise, detecção, alerta e possível contenção.

3 FORMAS DE ATAQUES E INVASÕES

O objetivo deste capítulo é apresentar de forma sucinta as principais formas de ataques e invasões e as respectivas vulnerabilidades que estas exploram. Dentre estas o *Denial of Service* (DoS) ou ataque de "negação de serviço", que é o tipo de ataque usado neste trabalho visando construir conhecimento sobre este tipo de vulnerabilidade.

3.1 A RAZÃO DE SER INVADIDO

De acordo com Santos (2005), qualquer código ou programa é passível de erros, pois, são projetados e escritos por seres humanos, estando então, sujeitos a falhas. Vulnerabilidades não se localizam somente nos códigos fonte dos programas, mas também, em sistemas mal configurados. Configurações padrões e sistemas desatualizados também geram muitas vulnerabilidades. Além disso, a falta de uma política de segurança também é falha grave no segmento de tecnologia da informação.

Ainda de acordo com Santos (2005), sistemas podem apresentar erros, e os mais graves são aqueles ligados a serviços de redes. Uma observação importante a ser feita em relação a segurança em sistemas de computador é que, deve-se manter em execução somente os serviços necessários, pois, quanto menos serviços ativos, menor serão os problemas de segurança.

Conforme Santos (2005), sistemas operacionais são passíveis de falhas e não há um que seja totalmente seguro. Cabe ao administrador consciente, implantar um sistema dentro dos padrões de segurança, usando para isso ferramentas disponibilizadas para este fim.

Assim sendo, para minimizar os incidentes de segurança, existem algumas recomendações básicas :

- Manter atualizados o sistema operacional e os aplicativos utilizados, principalmente os serviços de rede disponíveis, fundamentalmente atualizações pertinentes ao aspecto de segurança;
- Executar somente serviços necessários, desabilitando os desnecessários

e removendo-os da instalação se possível;

- Usar senha com poderes administrativos somente quando a tarefa a ser executada exija tal privilégio;
- Prover segurança física também é importante. Somente pessoas autorizadas devem ter acesso a servidores;
- Fazer uso de ferramentas de segurança. Em um servidor invadido, é grande o estrago causado (tempo sem o serviço, perda de dados, imagem afetada, etc). É importante coletar dados para uma posterior auditoria e deve-se reinstalar o servidor.

3.2 *PROBING*

Probing não é uma técnica propriamente dita, mas uma forma de obter informações sobre a rede ou sistema. As informações obtidas podem servir de base para uma possível invasão, com isso, pode-se utiliza-lo para descobrir quais serviços estão disponíveis na rede. As técnicas de realização de *probing* podem ser facilmente detectadas ou impedidas através de implementações eficientes no mecanismo de segurança na rede. (SANTOS, 2005)

3.3 *TROJAN HORSES*

São programas de computadores que realizam determinadas tarefas sem que seus usuários saibam, e que, normalmente afetam a segurança do sistema. Estas tarefas "obscuras" vão desde permitir que terceiros acompanhem ou tomem conhecimento do que se digita no teclado, como dados de acesso (senhas e *logins*), até abrir portas de acesso, gerando oportunidade para uma possível invasão. (SANTOS, 2005)

3.4 *BACKDOORS*

Normalmente um invasor procura garantir uma forma de retornar a um computador invadido, sem precisar utilizar-se dos métodos empregados na realização da invasão. Na maioria dos casos, o invasor quer retornar ao computador invadido sem ser notado. Este então, instala um programa no computador que

permitirá acesso a qualquer momento. (SANTOS, 2005)

O nome dado ao programa que abre esta porta para futuros acessos é *backdoor*. O *backdoor* permitirá acesso ao sistema mesmo que a falha que originou o primeiro acesso seja corrigida, isso é claro se o *backdoor* não for removido. Um *backdoor* pode ser escrito na forma de programa, *script*, ou até como uma série de procedimentos (criação de uma conta com acesso de administrador, etc), entre outras maneiras. (SANTOS, 2005)

3.3 *BUFFER OVERFLOW*

O *buffer overflow* é resultado do estouro de capacidade de armazenamento de um *buffer*. As ameaças deste tipo são consideradas críticas e bastante sérias e, apesar de muito conhecidas, originam-se exclusivamente na falta de conhecimento do programador no desenvolvimento seguro de um software. Vulnerabilidade que vem sendo amplamente utilizada para invasões remotas de computadores, onde o invasor consegue comprometer a estabilidade do sistema. Tecnicamente *buffer overflow* é um problema com a lógica interna do programa, mas a exploração dessa falha pode levar a sérios prejuízos. (SANTOS, 2005)

O objetivo da exploração contra um programa vulnerável a *buffer overflow* é conseguir que o invasor controle o programa atacado, e se este possuir privilégios suficientes, controlar a máquina (nível de administrador). (SANTOS, 2005)

3.4 *PASSWORD CRACKERS*

Password Crackers são programas desenvolvidos para quebrar senhas de usuários ou de programas. Este tipo de programa pode ser tanto usado para auxiliar o administrador à descobrir senhas fracas existentes no sistema, bem como ser usado por um invasor para descobrir senhas de usuários, e usá-las sem que o proprietário perceba. Em geral, *Password Crackers* tentam descobrir a senha do usuário através de um processo de força bruta. Comumente lentos, a eficiência dos *password crackers* depende inteiramente da qualidade das senhas. (SANTOS, 2005)

3.5 *SPOOFING*

A técnica de *spoofing* consiste na falsificação do remetente da mensagem de um pacote de transmissão de dados, para que o receptor o trate com se fosse de um outro utilizador. A técnica de *spoofing* também é utilizada por invasores para dificultar a descoberta da origem da invasão, uma vez que, as informações de origem dos pacotes são forjadas. (SANTOS, 2005)

Os principais e mais utilizados tipos de *spoofing* são : *IP Spoofing, ARP Spoofing e DNS Spoofing*. Como a técnica de *spoofing* constitui a falsificação de alguma informação, o *IP Spoofing* "falsifica" o IP, fingindo ser outro diferente do original. A técnica de *ARP Spoofing* consiste na falsificação do *MAC address*, redirecionando o tráfego para o impostor. No *DNS Spoofing*, o invasor compromete o servidor DNS e explicitamente altera as tabelas de endereço de IP – *hostname*, direcionando uma consulta DNS para outro servidor que não seja o "proprietário" do nome. (SANTOS, 2005)

3.6 ATAQUES DE NEGAÇÃO DE SERVIÇO DoS

De acordo com Santos (2005), os ataques conhecidos como DoS são caracterizados por uma tentativa explícita do atacante de impossibilitar que um usuário legítimo utilize determinado serviço. Ataques do tipo DoS geralmente não comprometem a privacidade dos dados. A finalidade deste tipo de ataque é tirar um serviço, servidor, computador ou até mesmo uma rede do ar. Ataques do tipo DoS muitas vezes são utilizados para "atrapalhar" ou desacreditar um serviço.

Ainda de acordo com Santos (2005), qualquer maneira de tirar um computador, serviço ou rede do ar pode ser considerado um ataque DoS. Como exemplo, pode-se citar um ataque com o intuito de gerar milhares de *logs* até encher o disco ou partição. Se o administrador colocar *logs* na mesma partição do sistema, tal ataque gerará um problema pois todo espaço do disco será ocupado.

Segundo Araújo e Assumpção (2010), ataques DoS podem ser podem ser realizados localmente ou remotamente, a saber :

- Localmente : ataques realizados quando o atacante tem acesso ao sistema. Esse acesso pode ser de forma física comprometendo a

integridade do *hardware* ou de forma computacional exaurindo os recursos de processamento e memória da máquina.

- Remotamente : ataques que inviabilizam o sistema a distância através de falhas dos serviços de rede sem necessariamente possuir uma conexão com o alvo.

Ainda segundo Araújo e Assumpção (2010), ataques DoS são realizados através de inúmeras formas. Ataques locais podem ser realizados através de um *buffer overflow* ou simplesmente desligando a fonte de alimentação do alvo. Todo ataque local exige que o atacante tenha acesso direto ao alvo, diferentemente dos ataques remotos onde a distância do alvo exige do atacante o uso de sólidos conhecimentos do conjunto de protocolos TCP/IP.

3.6.1 Pings e ICMP

O ICMP é um protocolo que pertence ao conjunto de protocolos TCP/IP, atuando na camada 3, sua função é verificar e diagnosticar problemas de rede. Sua arquitetura é apresentada na Figura 3.1.

	Bit 0-7	**Bit 8-15**	**Bit 16-23**	**Bit 24-31**
Cabeçalho IP **(160 bits ou 20 Bytes)**	Versão / DIH	Tipo de serviço	Comprimento	
	Identificação		*Bandeiras e offset*	
	Time To Live -TTL	Protocolo	Checksum	
	Endereço IP de origem			
	Endereço IP de destino			
ICMP *Payload* **(64+8+ bits ou bytes)**	Tipo de mensagem	Código	Checksum	
	Extinguir			
	Data (*opcional*)			

Figura 3.1 – Layout do pacote ICMP
Fonte : Santos, 2005, p.24

Segundo Araújo e Assumpção (2010), os pacotes ICMP trafegam sobre a forma de datagramas como os outros pacotes, e estes também estão sujeitos a erros. Além do cabeçalho IP o pacote ICMP contém um cabeçalho ICMP onde é especificado o tipo de mensagem, o código de retorno que identifica a resposta ao remetente indicando a situação encontrada. A tabela 3.1 mostra os tipos de mensagens ICMP.

Tabela 3.1 – Tipos de mensagens ICMP

Código do tipo	Mensagem ICMP
0	Resposta de echo (resposta ping)
3	Destino impossível de alcançar
4	Extinção de origem
5	Redirecionamento
8	Eco (solicitação ping)
11	Tempo de vida excedido
12	Problema de parâmetro
13	Solicitação de estampa de tempo
14	Retorno de estampa de tempo
17	Solicitação de máscara de endereço
18	Retorno de máscara de endereço

Fonte : Hatch et al, 2002, p.162

3.6.2 *Echo request* e *echo reply*

De acordo com Araújo e Assumpção (2010), quando utiliza-se o comando *ping* para verificar conectividade entre *hosts* dentro de uma rede, envia-se uma mensagem de requisição denominada *echo request*. Como resposta a requisição é retornada uma mensagem de confirmação chamada de *echo reply*. Esta mensagem retorna se houve resposta por parte do *host* que recebeu a requisição, caso não houver resposta, este informa o possível motivo da não alcançabilidade (*destination unreacheable*). A Tabela 3.2 mostra os possíveis sub-tipos de respostas.

3.6.3 Transformando um *ping* em arma

Segundo Araújo e Assumpção (2010), com envio de pacotes ICMP através

do *ping*, pode-se obter respostas sobre a conectividade de um *host* em uma rede, porém o uso indevido dessas requisições pode gerar gargalos na comunicação e consumo de recursos computacionais. A partir disso é possível levar servidores ao travamento (negação de serviço). Se um atacante enviar *pings* de tamanhos inválidos, pode obstruir o canal de comunicação ou, se conseguir gerar inúmeras requisições a um servidor forçando este a tentar responder a todas, pode inviabilizar os serviços deste.

Tabela 3.2 – *Destination unreachable* : possíveis sub-tipos de resposta ICMP

Code	Escription
1	*Network Unreachable error*
2	*Host Unreachable error*
3	*Protocol Unreachable error (the designated transpor is not supported)*
4	*Port Unreachable error (the designated protocol is unable to inform the host of the*
5	*Source route failed error*
6	*Destination network unknown error*
7	*Destination host unknown error*
8	*Source host isolated error (military use only)*
9	*The destination network is administratively prohibited*
10	*The destination host is administratively prohibited*
11	*The network is unreachable for type of servisse*
12	*The host is unreachable for type of servisse*
13	*Communication administratively prohibited (administrative filtering prevents packet*
14	*Host precedence violation (indicates the requested precedence is not permitted for*
15	*Precedence cutoff in effect (precedence of datagram is below the level set by the*

Fonte : Santos, 2005, p.26

3.6.4 Formas de ataques DoS

De acordo com Araújo e Assumpção (2010), ataques DoS ocorrem a partir de inúmeras formas. A seguir as formas de ataques mais conhecidas e aplicadas:

Ping of Death : uma das primeiras formas de ataque DoS, consiste no envio de pacotes com tamanhos que ultrapassam os 65536 bytes (o tamanho máximo permitido a um pacote). As redes atuais não são mais vulneráveis a esse tipo de ataque.

SYN *flood* : ataque baseado no estabelecimento de conexão do protocolo TCP, este chamado de *Tree-Way Handshake,* onde o cliente solicita a conexão através de um SYN e o servidor confirma a conexão mandando um SYN-ACK de

volta ao cliente, contudo este não envia o ACK para confirmar a conexão fazendo o servidor esperar um tempo limite (*time-out*), gerando assim um congestionamento na rede já que o ACK ausente na comunicação ocupa recursos do servidor (memória e processamento) e limita o uso de conexões ativas que um *software* pode realizar podendo limitar suas funcionalidades ou até travá-lo caso o número de conexões limites seja atingido.

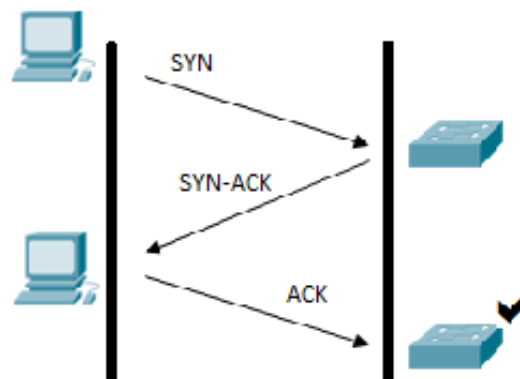



Figura 3.2 – Conexão normal cliente-servidor
Fonte : Elaborado pelo autor, 2013

Em uma conexão com SYN *flood* o atacante envia diversas vezes pacotes maliciosos com SYN e recebe respostas válidas, porém, não confirmando a conexão, deixando de enviar o ACK. O servidor é sobrecarregado e novas conexões não são realizadas com sucesso.

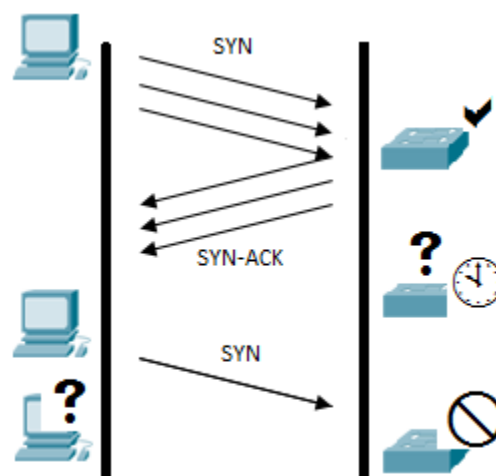



Figura 3.3 – Conexão com *SYN Flood*
Fonte : Elaborado pelo autor, 2013

Smurf : técnica conhecida como "reflexão" consiste na utilização de servidores de difusão (*broadcast*) para enviar requisições a todas as máquinas da rede. Assim o atacante envia uma solicitação a servidores *broadcast* com um IP falso fornecendo um endereço alvo fazendo com que todas as máquinas da rede reenviem uma resposta ao servidor de *broadcast* que repassa ao alvo congestionando assim toda solicitação ao mesmo.

UDP *flood* : técnica que utiliza o protocolo UDP para enviar requisições a um alvo. Diferentemente do SYN flood, esta técnica dispara requisições a todas as portas do alvo que por sua vez irá tentar identificar a qual processo essas requisições pertencem. Assim após a identificação o alvo tentará enviar uma mensagem ICMP ao destinatário que encontrou o destino da requisição enquanto continua sendo inundado com novas requisições até entrar em colapso.

Número da porta de origem (16 bits)	*Número da porta de destino (16 bits)*
Tamanho (16 bits)	*Soma de verificação (16 bits)*
Dados	

Figura 3.4 – Cabeçalho UDP
Fonte : Hatch et al, 2002, p.161

De acordo com Hatch et al (2002), como no TCP, o UDP usa endereço IP para enviar as informações ao sistema correto. O UDP usa números de porta para enviar os dados ao processo correto no sistema de destino. Já que o cabeçalho UDP não é verificado pelo sistema remetente, a porta de origem pode ser qualquer porta que um intruso quiser que seja. Já que nenhuma verificação de conexão é exigida para UDP, é muito fácil forjar tanto o endereço IP de origem quanto o número da porta UDP de origem.

3.7 DDoS

De acordo com Araújo e Assumpção (2010), o *Distribuited Denial of Service* (DDoS) é um ataque DoS amplificado. O atacante ao invés de usar uma única máquina para enviar um ataque, ele usa uma rede que pode conter até milhares de

máquinas para intensificar o seu ataque.

Segundo Jordão (2011), o ataque DDoS atinge sua meta excedendo os limites do servidor. Para tal, os responsáveis pelo ataque criam programas maliciosos que são instalados em diversas máquinas, as quais realizarão múltiplos acessos simultâneos ao site em questão.

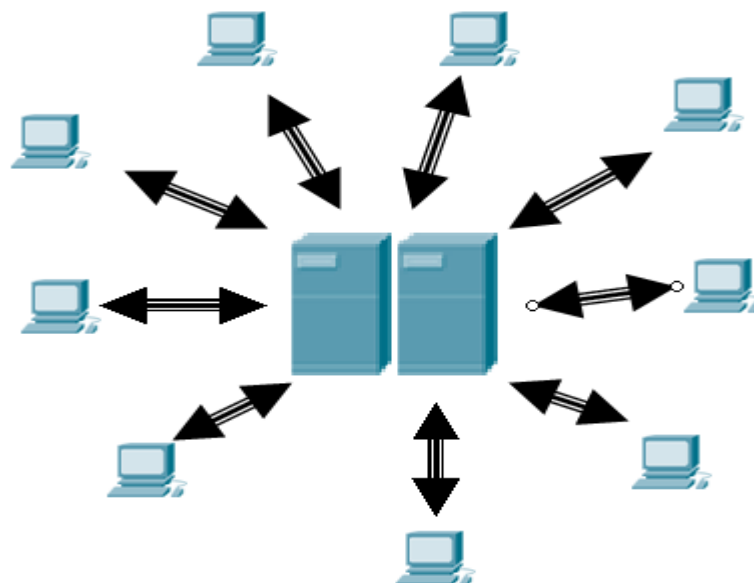



Figura 3.5 – Servidores são limitados também no envio e recebimento de dados
Fonte : Elaborado pelo autor, 2013

Ainda segundo Jordão (2011), como os servidores possuem limitações com relação ao número de acessos em um mesmo instante, acaba ocorrendo que o servidor não consegue atender as requisições e é retirado do ar. Um ataque DDoS pode simplesmente reiniciar os servidores ou pode causar o travamento do sistema que opera por trás do site.

Para aumentar a eficácia do ataque, um DDoS muitas vezes conta com a ajuda de máquinas zumbis, que integram uma rede zumbi. Computadores desse tipo foram infectados por pragas que tornam o acesso à Internet extremamente lento, isso porque eles estão sob o comando de outra máquina, também conhecido como computador-mestre. (JORDÃO, 2011)

E justamente por contar com uma legião de máquinas atacando é que o DDoS têm grande eficiência. Por se tratar de milhares de computadores realizando o ataque, fica muito mas difícil combatê-lo, porque os responsáveis pela segurança do servidor não conseguem estabelecer regras para impedir todos os acessos que

estão causando danos. (JORDÃO, 2011)

Para tentar combater um ataque DDoS, os profissionais que trabalham contra o ataque precisam efetuar configurações nos equipamentos que levam até o site desejado. Em geral, são utilizados filtros que vão determinar quais IPs podem acessar o site e quais são perigosos para o servidor. (JORDÃO, 2011)

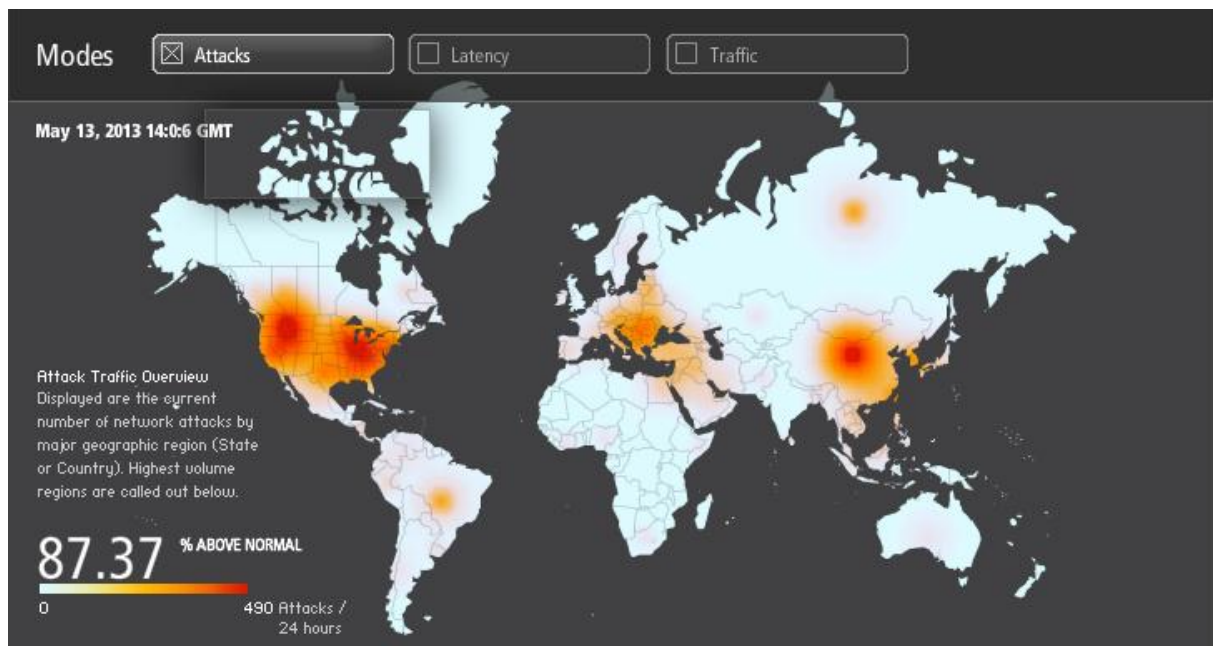



Figura 3.6 – Site Akamai, exibe a ocorrência de ataques DDoS em tempo real.
Fonte : Akamai, 2013

Outra solução é recorrer a empresas especializadas, como o Akamai, que utilizam diversos computadores para conter o ataque. Essa técnica é efetiva porque as máquinas estão em diferentes locais do planeta e combatem os zumbis dividindo a tarefa, de modo que cada computador de defesa combata um número reduzido de máquinas. (JORDÃO, 2011)

3.8 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo mostrar como funcionam os principais tipos de ataques e invasões e, mais especificamente o DDoS, e como este age na tentativa de sobrecarregar servidores visando a retirada do ar dos serviços destes.

4 TESTES E RESULTADOS

Neste capítulo mostrara-se como foi elaborada a aplicação dos testes, as ferramentas utilizadas para executar as simulações e coleta de dados, mostrar o ambiente criado para os testes, bem como os resultados esperados.

4.1 METODOLOGIA

Nesta pesquisa, serão considerados basicamente dois fatores como objeto de estudo e cada fator poderá assumir valores em duas situações conforme o injetor de pacotes usado nos testes.

Tabela 4.1 – Função das máquinas utilizadas na simulação

Fatores	Tempo de Ataque	
	BoNeSi	T50
Detecção de Intrusão - Gerando Alertas	*30 min.*	*30 min.*
Detecção de Intrusão - Gerando Alertas e Bloqueios	*30 min.*	*30 min.*

Fonte : Elaborado pelo autor, 2013

A carga de trabalho gerada para a realização dos testes é aplicada a partir de ferramentas que possibilitam realizar *Stress Testing* em rede locais. São eles, os injetores de pacotes BoNeSi e o injetor T50, que permitem realizar uma simulação de ataque DDoS como se fossem máquinas zumbis realizando ataque simultaneamente, como ilustra a Figura 4.1.

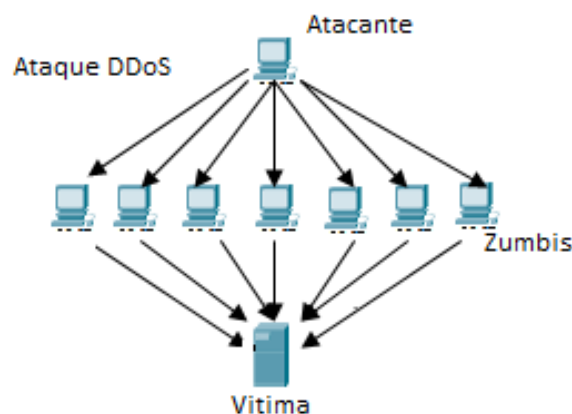



Figura 4.1 – Ataque DDoS.
Fonte : Elaborado pelo autor, 2013

O principal propósito de um ataque DDoS é inviabilizar um serviço, ou seja, tirar determinado serviço do ar, porém, o objetivo dos testes aqui aplicados é de avaliar o impacto da contenção realizada pelo IDS em conjunto com o *firewall*.

4.2 FERRAMENTAS UTILIZADAS

Algumas ferramentas foram usadas para adicionar funcionalidades ao *Snort* ou trabalhar em conjunto com ele. Algumas bibliotecas também são mencionadas por serem necessárias ao funcionamento de uma ou mais ferramentas.

Todas as ferramentas que foram utilizadas na elaboração deste trabalho são de uso gratuito, disponíveis para *download* e sem restrições de uso.

4.2.1 MYSQL

Conforme Lari e Amaral (2004), o MySQL é o banco de dados relacional de código aberto, escrito em C e C++, mais utilizado em todo mundo. Suas maiores virtudes são a velocidade, a confiabilidade e a portabilidade. Pode lidar com grandes bases de dados e já foi testado com mais de 50 milhões de registros.

```
mysql> show tables;
+--------------------+
| Tables_in_snort_db |
+--------------------+
| acid_ag            |
| acid_ag_alert      |
| acid_event         |
| acid_ip_cache      |
| base_roles         |
| base_users         |
| data               |
| detail             |
| encoding           |
| event              |
| icmphdr            |
| iphdr              |
| opt                |
| reference          |
| reference_system   |
| schema             |
| sensor             |
| sig_class          |
| sig_reference      |
| signature          |
| tcphdr             |
| udphdr             |
+--------------------+
```

Figura 4.2 – Tabelas criadas no MySQL e usadas pelo *Snort*.
Fonte : Elaborado pelo autor, 2013

Neste trabalho, o MySQL funcionará como banco de dados onde o *Snort* armazenará seus alertas.

4.2.2 PHP

De acordo com Lari e Amaral (2004), o *HyperText Preprocessor* (PHP) é uma linguagem de *script* de código livre muito utilizada pela comunidade de *software* livre. Esta linguagem visa o desenvolvimento de aplicações para internet dentro do HTML, ou seja, em um código HTML pode-se inserir linhas de código PHP para realizar determinada tarefa, como, coletar dados de formulários, enviar *cookies*, entre outros.

4.2.3 BASE

Neste trabalho, o PHP precisa ser instalado, pois o *Basic Analysis and Security Engine* (BASE) se baseia nesta linguagem.

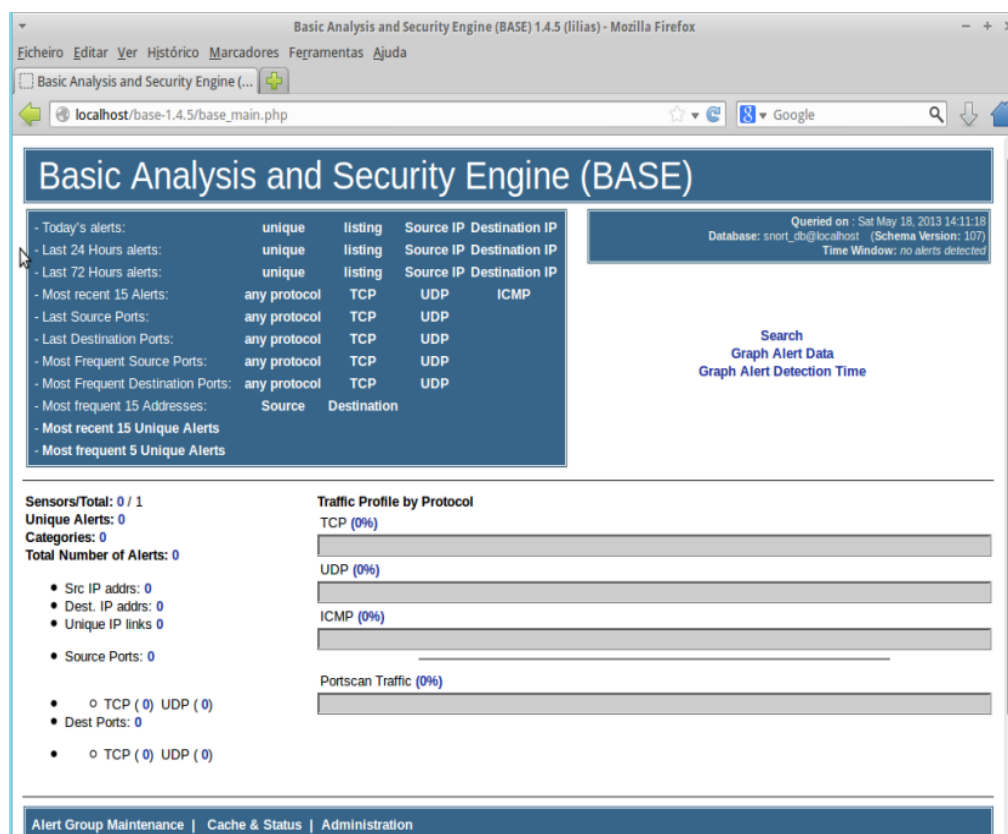



Figura 4.3 – BASE Ferramenta estatística para analise dos dados do *Snort*.
Fonte : Elaborado pelo autor, 2013

Esta ferramenta apresenta estatísticas de alertas, permitindo descobrir o número de acessos divididos por protocolo, por portas de origem e destino, entre

outras informações. Cada pacote individual registrado pode ser exibido em um formato decodificado, mostrando várias *flags*, opções e conteúdos dos pacotes.

4.2.4 APACHE

O uso do BASE neste trabalho fornece uma interface Web para um acompanhamento dos eventos ocorridos em nosso ambiente de teste e gerados pelo *Snort*. Construída em PHP, é uma importante ajuda para acompanhamento dos resultados e auditoria de dados.

Neste contexto encontramos o Apache. O Apache é o servidor de web mais utilizado em todo mundo. É mantido pela "*The Apache Software Foundation*".

4.2.5 OINKMASTER

Como já relatado no capítulo sobre o *Snort*, este realiza a detecção de tentativas de invasão através da comparação dos pacotes capturados com sua base de dados de regras. A atualização destas regras se dá pela adição de regras novas ou por modificação das regras antigas. Existem várias fontes de regras para Snort, sendo a principal delas localizada no site oficial do *Snort*, www.snort.org.

O oinkmaster é uma ferramenta desenvolvida em *Perl* para automatizar o processo de *download* e mesclagem das regras do *Snort*.

4.2.6 LIBPCAP

A Libpcap é uma biblioteca de funções que ajuda o programador a realizar tarefas relacionadas com redes de computadores. No lugar de programar com *sockets*, por exemplo, o programador pode utilizar funções já definidas, como captura de pacotes diretamente da placa de rede. (LARI e AMARAL, 2004)

4.2.7 ADODB

De acordo com Lari e Amaral (2004), a Adodb é uma biblioteca de classes utilizada por banco de dados que facilita o trabalho do programador no sentido da flexibilidade. Normalmente, um programa em PHP possui funções que chamam

determinado banco de dados.

A Adodb suporta vários banco de dados, entre eles, MySQL, PostgreSQL, Interbase, Firebird, Informix, Oracle, MS SQL 7, Foxpro, *Access*, ADO, Sybase e outros. Usa-se essa biblioteca neste trabalho, para conectar o MySQL junto com o *Snort*.

4.2.8 *GUARDIAN*

O *Snort* é uma ferramenta que gera alertas das tentativas de invasão. O *Snort* sozinho não faz o bloqueio automático de ataques. Para esta tarefa ou compila-se o *Snort* com suporte ao módulo *inline* ou utiliza-se uma ferramenta extra para a realização deste bloqueio. Neste trabalho optou-se pela utilização da ferramenta chamada *Guardian*.

O *Guardian* tem a capacidade de ler os logs do *Snort* em tempo real, agindo contra um possível invasor no momento da tentativa. O *Guardian* apresenta scripts para realizar bloqueios compatíveis com os *firewalls Iptables, Ipchain e Ipfw*.

4.2.9 *WIRESHARK*

O wireshark, antigamente conhecido como Ethereal, é um programa que analisa o tráfego de rede, e o organiza por protocolos. As funcionalidades são parecidas com o tcpdump mas com interface *Grafic User Inferface* (GUI), com mais informações e com a possibilidade de uso de filtros.

É então possível controlar o tráfego de uma rede e saber tudo o que entra e sai do computador, em diferentes protocolos, ou da rede a qual o computador está ligado. Neste trabalho, este programa será usado analisar o tráfego do *host* antes e sob ataque DDoS, afim de verificar os bloqueios de dados e tráfego de rede.

4.2.10 T50

O T50 é uma ferramenta desenvolvida por um brasileiro, Nelson Brito, capaz de usar uma forma de teste para determinar a estabilidade de um determinado sistema. Trata-se de testar além da capacidade operacional normal, muitas vezes a um ponto de ruptura, afim de avaliar os resultados.

O T50 suporta até 15 protocolos e pode atingir até 120.000 pacotes por segundo em redes 100BaseT, e 1.000.000 de pacotes por segundo em redes 1000BaseT. Neste trabalho o T50 será usado em nossos testes simulando um ataque do tipo DDoS.

O T50 pode ser encontrado no link http://t50.sourceforge.net/.

4.2.11 BONESI

Projetado para estudar o efeito de ataques DDoS, o BoNeSi é uma ferramenta criada para simular ataque de Botnet em um ambiente de rede. O BoNeSi gera inundações de pacotes ICMP, UDP e TCP (HTTP) de tamanho definido, a partir de diferentes endereços IP.

Primeira ferramenta criada para simular inundações HTTP a partir de redes de bot em larga escala, o BoNeSi também tenta evitar gerar pacotes identificáveis com padrões simples (que podem ser facilmente filtrados).

O BoNeSi pode ser encontrado no link http://code.google.com/p/bonesi/.

4.2.12 VIRTUALBOX

Software que permite que um único *host* possa criar e executar um ou mais ambientes virtuais, o Oracle VirtualBox é um *software* de virtualização de uso gratuito para uso doméstico ou em empresas, e também tem seu código disponível sob os termos da GNU *General Public Licence* (GLP).

Neste trabalho, o VirtualBox será usado para criar-se as VMs que contém os programas implementados, assim como, todo o ambiente usado para testá-los. O virtualBox pode ser encontrado no site www.virtualbox.org.

4.3 AMBIENTE DE TESTES

O ambiente de testes utilizado nesta avaliação de desempenho foi um ambiente de rede local. O aplicativos T50 e Bonesi, utilizados para injeção de pacotes, embora simulem um ataque DDoS, e possam ser modificados para uso na rede externa, são compilados para uso somente em rede local, com o objetivo de

fazer *stress testing* na rede, com objetivo de avaliar desempenho. Seus autores assim indicam o uso destes programas.

O ambiente de rede local foi construído através de software de virtualização VirtualBox. A máquina hospedeira possui as seguintes configurações de *hardware* e Sistema Operacional (SO) : Processador Intel® Core™ i7-2600 3.40 GHz, memória RAM 4,00 GB, 1 TB Hard Disk (HD) e o SO Windows 7 Professional 64 bits.

Nas *Virtual Machines* (VM) ou máquinas virtuais, foram utilizadas as seguintes configurações de *hardware* e SO : memória RAM 1,5 MB, 30MB HD e SO XUbuntu versão 12.10. A interface de rede foi configurada como “modo *bridge*” e usa comunicação padrão *Ethernet* 100BASE-TX.

A Figura 4.4 ilustra o ambiente geral dos testes utilizando as VMs que são compostas pela Virtual-VM1, que representa o servidor de segurança, onde está instalado o *Snort* e demais aplicativos, e onde serão feitas as contenções (bloqueios) no nível em que o *Guardian* estiver habilitado. E mostra também as demais máquinas, que são as VMs que contém os injetores de pacotes e de onde se originam os ataques.

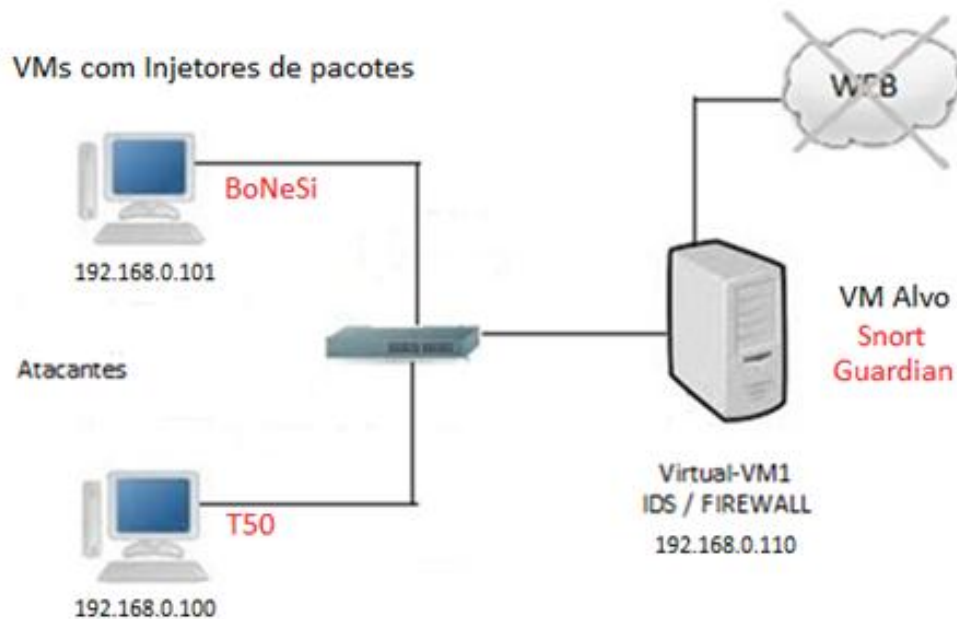



Figura 4.4 – Ambiente de Testes
Fonte : Elaborado pelo autor, 2013

Vale ressaltar que a máquina hospedeira, e consequentemente, as VMs, não estavam com conexão à Internet.

4.4 RESULTADOS EXPERIMENTAIS

Para a obtenção dos resultados dos experimentos nestes testes, foram realizadas quatro simulações de ataques do tipo DDoS, a partir dos injetores de pacotes.

Duas simulações foram feitas a partir do injetor de pacotes T50, sendo que na primeira, detecção, o objetivo era avaliar o número de alertas gerados a partir das tentativas de intrusão, tendo suas correspondências no banco de dados de regras do *Snort*, estando o *Guardian* nesta fase, desabilitado. Na segunda simulação, detecção com bloqueio, o objetivo é avaliar os bloqueios criados no firewall do Linux (*iptables*), a partir dos alertas gerados pelo *Snort*. Fazendo assim, com que se verificasse a efetiva contenção das tentativas de ataque. Nesta fase o *Guardian* encontrava-se habilitado.

Outras duas simulações de ataques com as mesmas características e objetivos das citadas acima, foram realizadas utilizando-se desta vez, o injetor de pacotes BoNeSi.

As máquinas virtuais alvos dos ataques, uma para cada simulação, foram clonadas a partir de uma máquina virtual "base", com todos os aplicativos instalados e configurados, porém sem nenhum registro resultante de qualquer ataque ou captura de pacotes, para que não houvesse influência no resultado final de cada teste. Uma máquina virtual foi usada individualmente para cada injetor de pacotes, embora não houvesse razão para usar-se uma única VM com os dois programas injetores instalados.

4.5 SIMULAÇÃO DE ATAQUE COM O INJETOR T50

Na primeira simulação usando o injetor de pacotes T50, realizou-se o ataque ao *Host* com *Snort,* sem iniciar o serviço do *Guardian*. Assim sendo, o *Snort* analisou os pacotes que correspondiam a assinaturas de ataques contidas em seu banco de dados de regras, gerando os respectivos alertas. O *Snort* retornou os resultados como mostra a Figura 4.5.

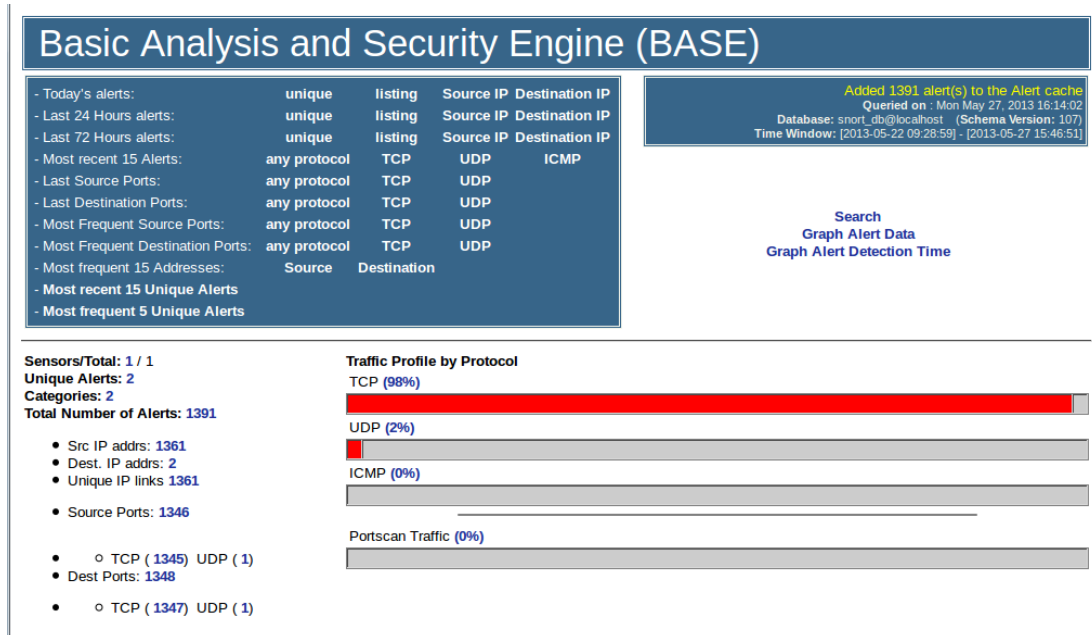



Figura 4.5 – Tela do BASE, ataque do T50 com *Guardian* desabilitado
Fonte : Elaborado pelo autor, 2013.

O T50 foi executado a partir da seguinte linha de comando :

```
./t50 192.168.0.110 --flood --turbo
```

Onde, 192.168.0.110, corresponde ao IP do *host* a ser atacado. O parâmetro *flood* refere-se ao número de pacotes enviados, no caso, ilimitado (podendo atingir até 120.000 pacotes no caso da rede 100BaseT), e o parâmetro *turbo* é usado para intensificar o ataque.

Quadro 4.1 – Protocolos usados no ataque com T50

eth : 1387 pacotes analisados (100%)		
Protocolos	**Pacotes**	**Percentual**
IPv4	1363	98,270%
UDP	3	0,216%
TCP	1360	98,053%
IPv6	20	1,442%
UDP6	11	0,793%
ARP	4	0,288%

Fonte : Elaborado pelo autor, 2013

O Quadro 4.1 mostra os protocolos dos pacotes analisados pelo snort no tempo de ataque com T50.

Nos gráficos exibidos nas figuras 4.6 e 4.7, verifica-se o tráfego de rede nos

momentos das simulações de ataques (dados em milhares de pacotes por segundo, em intervalos de 10s). Observa-se que o tráfego se mantém semelhante nas duas simulações, porém nota-se um pico maior ocorrendo a partir dos 1800s, momento em que foi finalizada a execução do injetor de pacotes. Pode-se observar este aumento de tráfego nos 150s seguintes, e sendo maior, na simulação em que foi realizado a detecção com bloqueio.

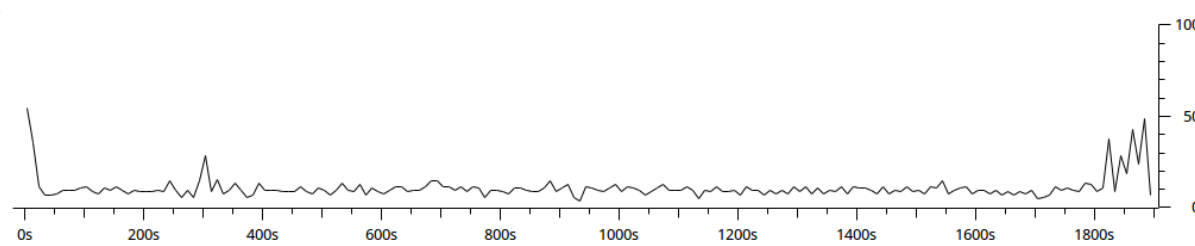



Figura 4.6 – Tráfego de pacotes na rede durante ataque. detecção T50
Fonte : Elaborado pelo autor, 2013

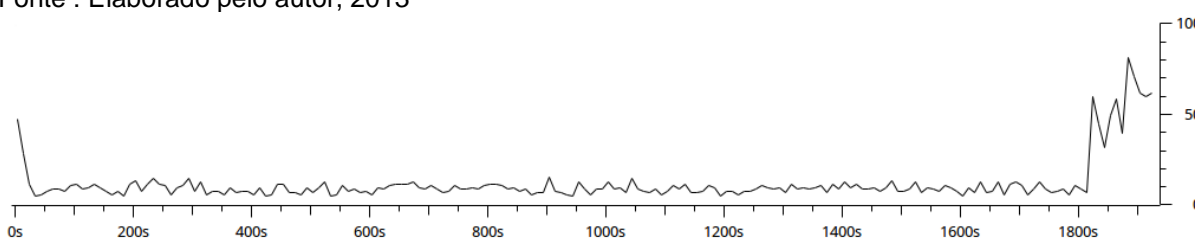



Figura 4.7 – Tráfego de pacotes na rede durante ataque. detecção com bloqueio T50
Fonte : Elaborado pelo autor, 2013

Quadro 4.2 – Resultados gerados pelo *Snort* nas simulações com T50.

	Detecção		Detecção e Bloqueio	
Pacotes Recebidos	50.989.996		49.609.890	
Pacotes Descartados	50.988.338		49.607.817	
Pacotes Analisados	1658	0,003% / recebidos	2073	0,004% / recebidos
Pacotes por minuto	53		64	
Alertas Gerados	1442	86,972% / analisados	1429	68,934% / analisados
Tempo de Execução	1887,395s		1934,327s	

Fonte : Elaborado pelo autor, 2013

Como mostrado no Quadro 4.2, na linha **Pacotes Analisados**, observa-se um maior número de pacotes na simulação de detecção com bloqueio.

A Figura 4.8 mostra o percentual de utilização da CPU pelo processo do *Snort*, onde compare-se as simulações de somente detecção e detecção com bloqueio. Durante o tempo de simulação de ataque (30min.), observa-se que não há

grande variação do uso da CPU na comparação entre as duas simulações.

Os dados foram obtidos a partir do uso do comando ***top*** *do Linux*, que fornece uma visão dinâmica dos processos que estão sendo executados, monitorando, entre outros dados, o percentual de utilização do processador por cada processo.

A seguinte sintaxe foi utilizada para registrar os dados :

```
#top -b -n 62 -d 30 -u snort > teste.txt
```

Onde, **-b** gera processamento em lote, **-n** refere-se ao número de repetições das verificações, **-d** é o intervalo de tempo (em segundos) entre cada verificação, **-u** exibir somente o processo S*nort*, enviando o resultado para um arquivo texto.

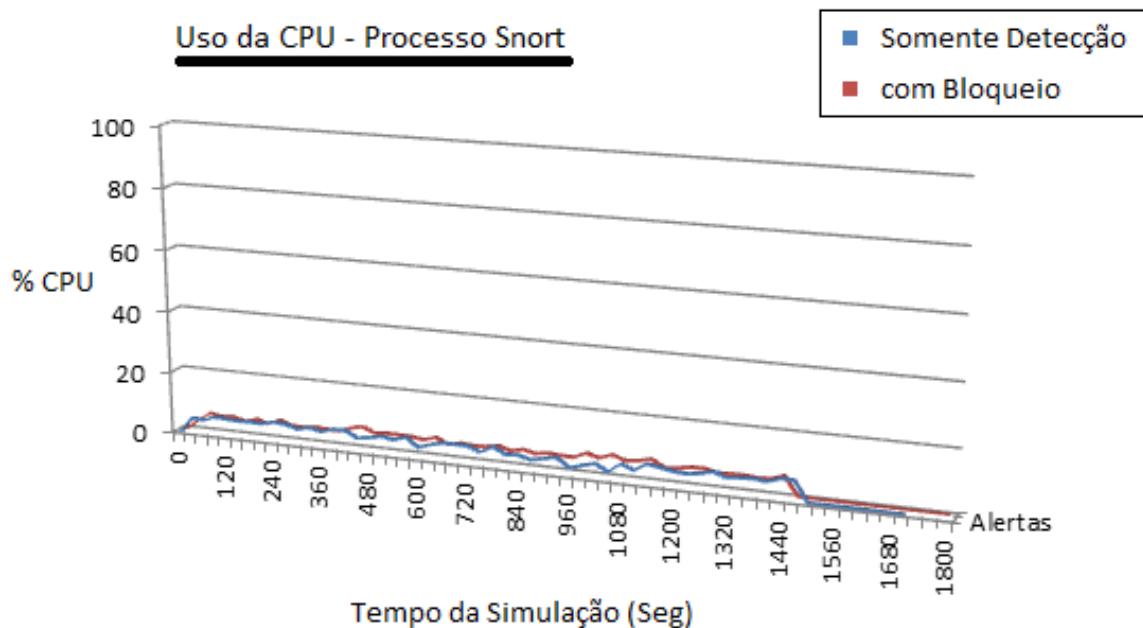



Figura 4.8 – Uso da CPU pelo processo *Snort* na simulação com T50
Fonte : Elaborado pelo autor, 2013

A Figura 4.9 mostra parte do relatório de alertas gerados pelo BASE, onde observa-se as informações como assinatura da invasão, os endereços de IP origem (atacante) e destino (alvo), protocolo e porta utilizada no ataque.

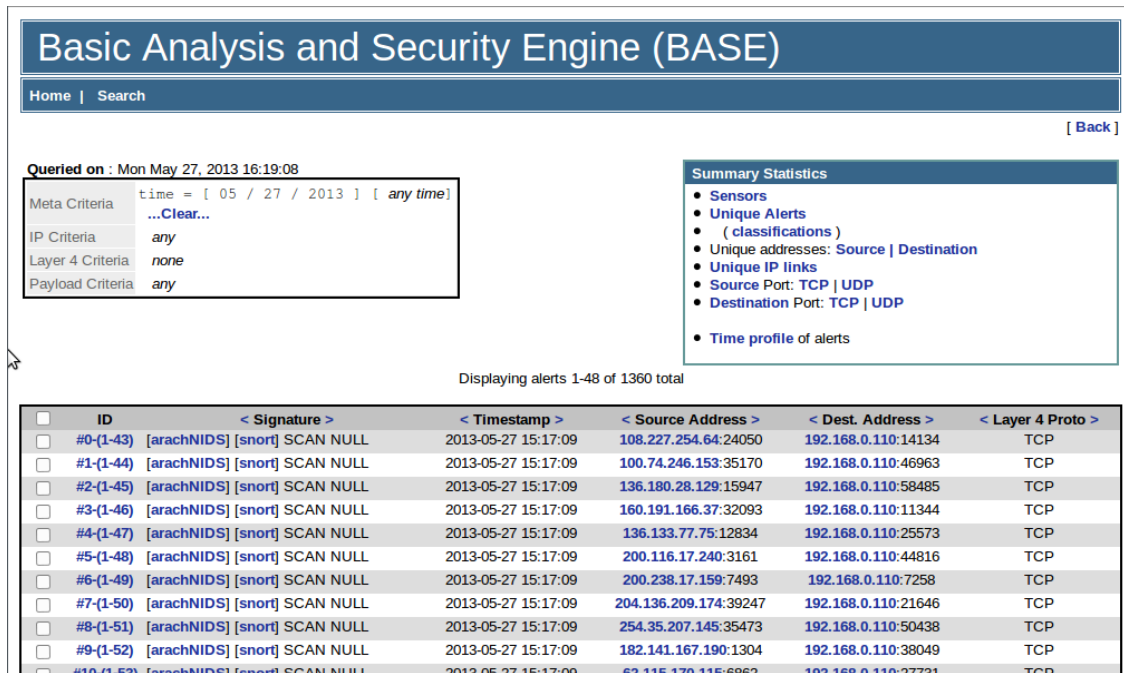



Figura 4.9 – Relatório de Alertas do BASE na simulação com T50
Fonte : Elaborado pelo autor, 2013

Na simulação com T50 foram gerados 1429 alertas, e a partir destes, foram criadas 1387 regras no *firewall*. A Figura 4.10 mostra parte das regras criadas no *iptables*, sendo todas elas na *chain INPUT*.

Figura 4.10 – Bloqueios feitos no *iptables* na simulação com T50.
Fonte : Elaborado pelo autor, 2013

4.6 SIMULAÇÃO DE ATAQUE COM O INJETOR BONESI

Assim como feito com o injetor T50, duas simulações foram realizadas com o injetor de pacotes Bonesi, avaliando-se os mesmos itens das simulações anteriores.

O Bonesi foi executado a partir da seguinte linha de comando:

```
bonesi -i 50k-bots 196.168.0.110:80
```

Onde, o parâmetro -i refere-se ao arquivo que contem uma lista de IPs válidos que farão o papel dos IPs origem, ou seja, de onde partem os ataques (máquinas zumbis). O Bonesi conta com um arquivo, o 50k-bots, que possui uma lista com 50.000 endereços de IPs válidos usados como atacantes. E finalmente, os dois últimos parâmetros referem-se ao IP e porta destino, ou alvo do ataque.

Nos gráficos exibidos nas Figuras 4.11 e 4.12, verifica-se que o tráfego da rede nos momentos dos ataques tem maior oscilação entre as duas simulações do que a constatada nas simulações com o injetor T50. Aqui observa-se também, um pico maior ocorrendo a partir dos 1800s, momento em que foi finalizada a execução do injetor de pacotes. Pode-se observar este aumento de tráfego nos 150s seguintes, e no caso da simulação com o Bonesi, este pico é maior na simulação em que foi realizada somente detecção, ao contrário do que observou-se nas simulações com o T50.

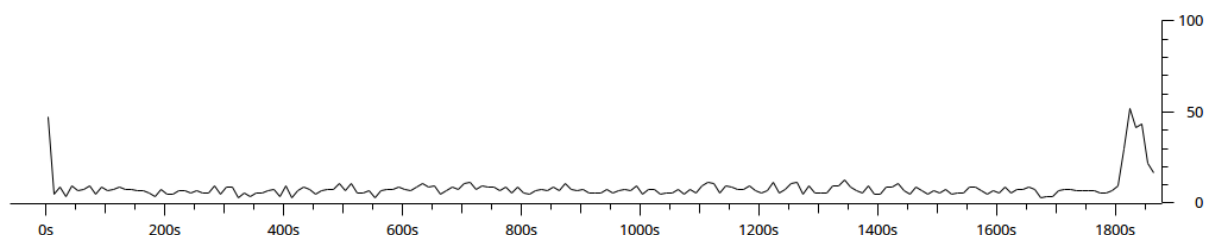



Figura 4.11 – Tráfego de pacotes durante ataque – detecção Bonesi.
Fonte : Elaborado pelo autor, 2013

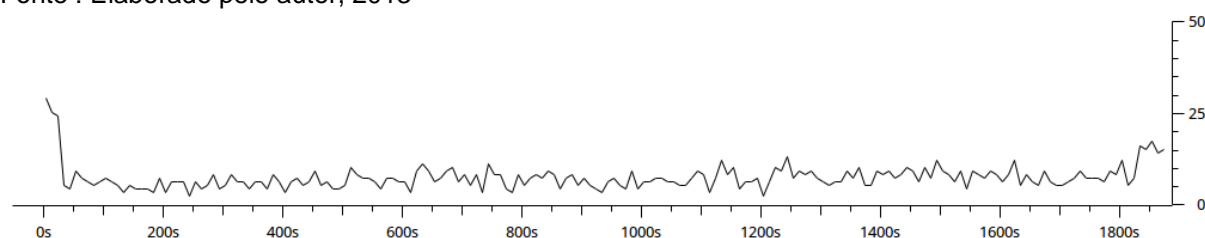



Figura 4.12 – Tráfego de pacotes durante ataque - detecção bloqueio Bonesi.
Fonte : Elaborado pelo autor, 2013

Nas simulações usando o injetor Bonesi, foram constatados resultados bem

distintos, principalmente no que se refere aos pacotes analisados e os alertas gerados. Comparando-se estas simulações com as feitas usando o T50, observa-se um número de pacotes analisados bem maior, ao passo que o número de alertas gerados foi bem menor que nos números constatados com o T50. Pode-se verificar estes números no Quadro 4.3, nas linhas **Pacotes Analisados** e **Alertas Gerados**. Observa-se também um consequente aumento no número de pacotes analisados por minuto.

Quadro 4.3 – Resultados gerados pelo *Snort* nas simulações com Bonesi.

	Detecção		Detecção e Bloqueio	
Pacotes Recebidos	36.184.131		49.055.322	
Pacotes Descartados	36.177.109		49.045.784	
Pacotes Analisados	7022	0,019% / recebidos	9538	0,019% / recebidos
Pacotes por minuto	292		307	
Alertas Gerados	17	0,242% / analisados	17	0,178% / analisados
Tempo de Execução	1847,395s		1864,277s	

Fonte : Elaborado pelo autor, 2013

A utilização da CPU pelo processo *Snort* também foi maior na simulação com o injetor Bonesi, como mostrado na Figura 4.13.

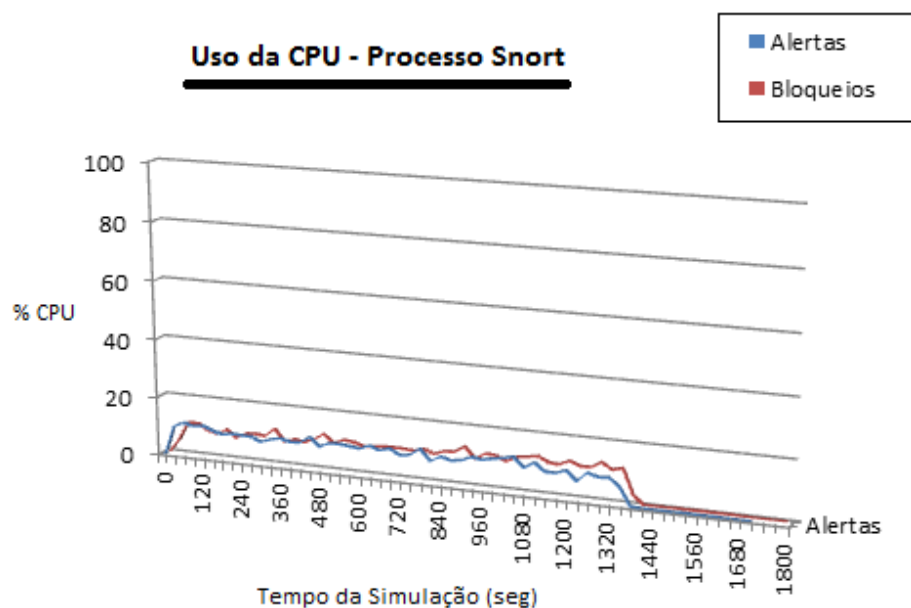



Figura 4.13 – Uso da CPU pelo processo *Snort* na simulação com Bonesi.
Fonte : Elaborado pelo autor, 2013

4.7 COMPARAÇÃO ENTRE OS INJETORES

Como mencionado anteriormente, foi usado nas simulações um tempo de ataque de trinta minutos. Nesta amostragem de tempo, o *Snort* acusou um número de pacotes recebidos que não apresenta grandes variações entre as quatro simulações, como mostra a Figura 4.14.

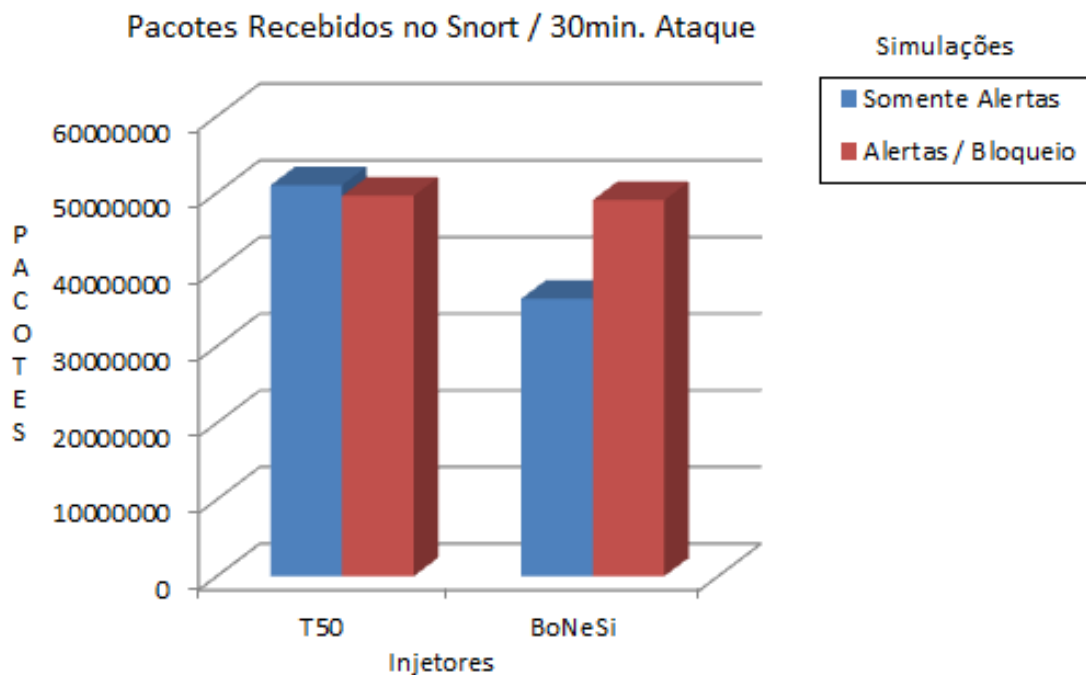



Figura 4.14 – Comparação entre os injetores dos pacotes recebidos.
Fonte : Elaborado pelo autor, 2013

Na comparação entre os dois injetores de pacotes, em ambas as situações, ou seja, somente alertas e gerando bloqueios (contenções), constata-se que nas simulações usando o Bonesi, o número de pacotes analisados é bem maior do que o observado nas simulações com o T50. Porém, observa-se que o número de pacotes analisados não corresponde necessariamente a um número maior na incidência de alertas. Pois, nas simulações com o T50, o número de alertas gerados foi baixo em relação ao número maior de pacotes analisados, como mostrado nos gráficos da figura 4.15.

Conclui-se nesse caso que, na simulação com o injetor T50, embora um número menor de pacotes tenha sido analisado, entre eles encontra-se um maior número que tem correspondências nas assinaturas constantes no banco de dados

do *Snort*, ou seja, entre os pacotes analisados um maior número de pacotes que contém código malicioso.

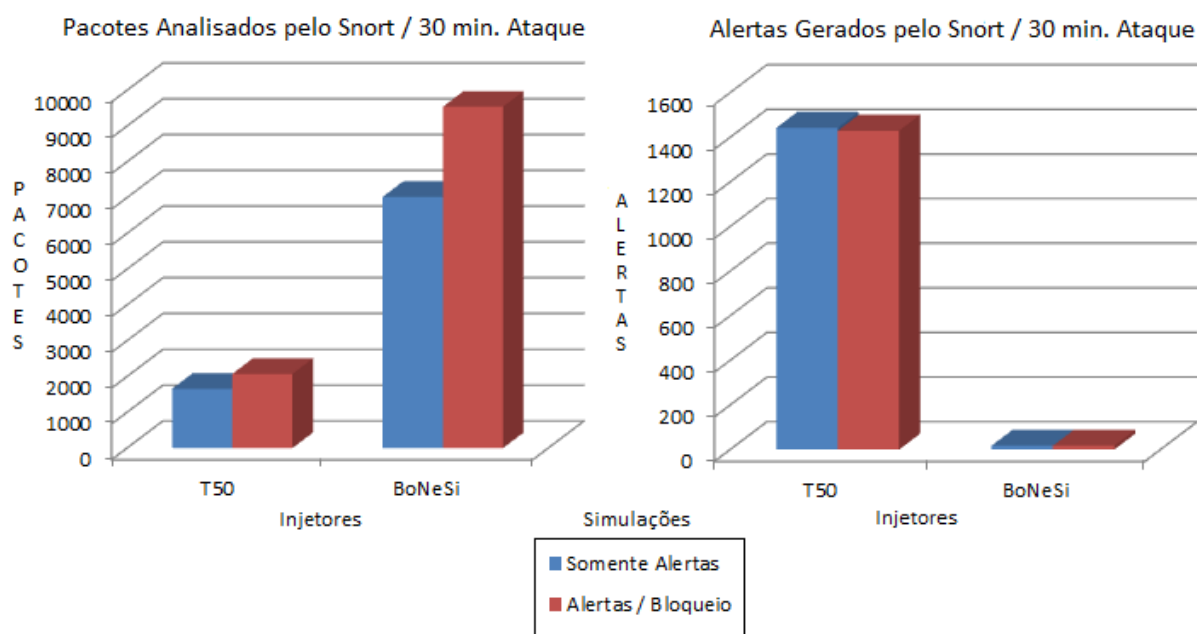



Figura 4.15 – Pacotes Analisados x Alertas/Bloqueios
Fonte : Elaborado pelo autor, 2013

Uma constatação feita, na simulação de detecção e bloqueio usando o injetor Bonesi, em que somente 17 alertas foram gerados, nenhuma regra de bloqueio foi criada no *firewall* (*iptables*), ou seja, os pacotes atacantes passaram pelo servidor de segurança.

Nas situações em que o objetivo era a realização de contenção, na simulação com o injetor Bonesi, e constatando-se o grande número de pacotes analisados pelo *Snort*, com um pequeno número de alertas gerados, constata-se que a maior parte dos pacotes com conteúdo malicioso injetado pelo Bonesi, não encontrou correspondência nas assinaturas de tentativas de intrusão contidas no banco de dados do *Snort*, ou seja, este injetor tem maior capacidade de burlar o IDS, considerando-se as regras disponibilizadas que nele estão contidas.

Constatou-se também que a contenção de ataques com o uso das ferramentas *Snort/Guardian* foi feita, porém sua eficiência, ou a falta dela, está relacionada à constante manutenção do conjunto de regras do *Snort*.

4.8 DIFICULDADES ENCONTRADAS

Algumas dificuldades foram encontradas no desenvolvimento do trabalho:

- O uso dos injetores no ambiente virtual não possibilitou um resultado realista.
- Os injetores usados no ambiente de maquinas virtuais causava travamento na maquina alvo durante as simulações, impedindo um melhor acompanhamento, sé possibilitando colher resultados após a finalização do injetor.

4.9 TRABALHOS FUTUROS

Entre os possíveis trabalhos futuros pode-se indicar:

- Uma comparação entre as formas *inline* do *Snort* e a optada por este trabalho usando o *guardian* como ferramenta adicional para realizar contenção;
- A construção de um injetor de pacotes, o que permite construir conhecimento mais detalhado sobre os ataques
- Criar regras no *Snort* específicas para ataques DoS;
- Configuração do *Snort* em plano, que consiste em uma variante do Snort em que as regras são criadas a partir do conhecimento da rede local.

CONCLUSÃO

A implementação de um IDS composta neste trabalho, foi realizada para mostrar como pode-se criar uma camada extra de segurança na rede local ou em *host*, possibilitando, além da construção de conhecimento sobre segurança da rede local, a capacidade de realizar a contenção das tentativas de violação da segurança.

O uso do IDS constitui fundamental ferramenta de auxílio na manutenção da segurança e seu grau de eficiência está intimamente ligado ao conjunto de informações construído sobre a rede local e do conhecimento inerente a redes de

computadores. E, ao mesmo tempo em que constitui ferramenta de segurança, o IDS bem utilizado, é também, fonte de constante informação e aprendizado sobre tráfego de dados, aperfeiçoando o nível de segurança do ambiente onde está implantado.

A principal contribuição deste trabalho foi na construção de conhecimento acerca da implantação um sistema de detecção de intrusão. Coletar dados a respeito do comportamento do IDS quando sobre ataques e avaliar sua eficiência trabalhando em conjunto com outras ferramentas para realização da contenção destes ataques.

REFERÊNCIA BIBLIOGRÁFICA

AKAMAI. **Real time web monitor**. Akamai Technologies, 2013. Disponível: http://www.akamai.com/html/technology/dataviz1.html

ARAÚJO, FELIPE GUSTAVO SOUZA; ASSUMPÇÃO, MARCELO HOOVER PIMENTEL. **SIGIN – SISTEMA PARA GERENCIAMENTO DE DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO** . 2010. Trabalho Final de graduação (Bacharelado em Informática) – Instituto de Ciências Exatas e Tecnológicas, Universidade Católica do Salvador - UCSAL, Bahia

CASWELL, BRIAN at al. **Snort 2 – Sistema de Detecção de Intruso - *Open Source***. Rio de Janeiro : Alta Books, 2003.

HATCH, BRIAN; LEE JAMES; KURTZ GEORGE **HACKERS Linux EXPOSTOS – Segredos e Soluções Para a Segurança do Linux**. São Paulo : Makron Books, 2002.

JORDÃO, FABIO **DDoS: como funciona um ataque distribuído por negação de serviço.** 2011. Disponível em : http://www.tecmundo.com.br/segurança/10970-ddos-como-funciona-um-ataque-distribuido-por-negacao-de-servico.htm Acesso em 10 Abril. 2013.

LARI, PAULO AUGUSTO MODA; AMARAL, DINO MACEDO **Snort, MySQL, Apache e ACID.** Rio de Janeiro : Brasport, 2004.

LIEIRA, JULIO FERNANDO **Segurança de Redes.** 2012, 87p. Apostila Segurança de Redes de Computadores, Fatec Lins, Lins – SP.

NOBRE, J.C.A.. **Ameaças e Ataques aos Sistemas de Informação : Prevenir e Antecipar** : Cadernos UniFOA, Volta Redonda, ano 2, nº. 5, dez 2007. Disponível : http://www.unifoa.edu.br/pesquisa/caderno/edicao/05/11.pdf

SANTOS, B.R. **DETECÇÃO DE INTRUSOS UTILIZANDO O SNORT** . 2005. TCC. (PÓS GRADUAÇÃO EM ADMINISTRAÇÃO DE REDES) - Departamento de Computação, Universidade Federal de Lavras, Minas Gerais

ANEXO A – Instalação e Configuração

Neste anexo mostramos a instalação e configuração dos aplicativos utilizados no desenvolvimento deste trabalho. Ao longo do capitulo 4, onde são descritas estas ferramentas, constam o site ou link onde elas podem ser encontradas, entretanto, outros mais serão referenciados neste anexo.

Criando e instalando as máquinas virtuais

As VMs criadas para os testes, usaram todas, as mesmas configurações e sistema operacional, portanto, para a VM usada como servidor de segurança e para as demais VMs onde são instalados os injetores, o mesmo procedimento será usado.

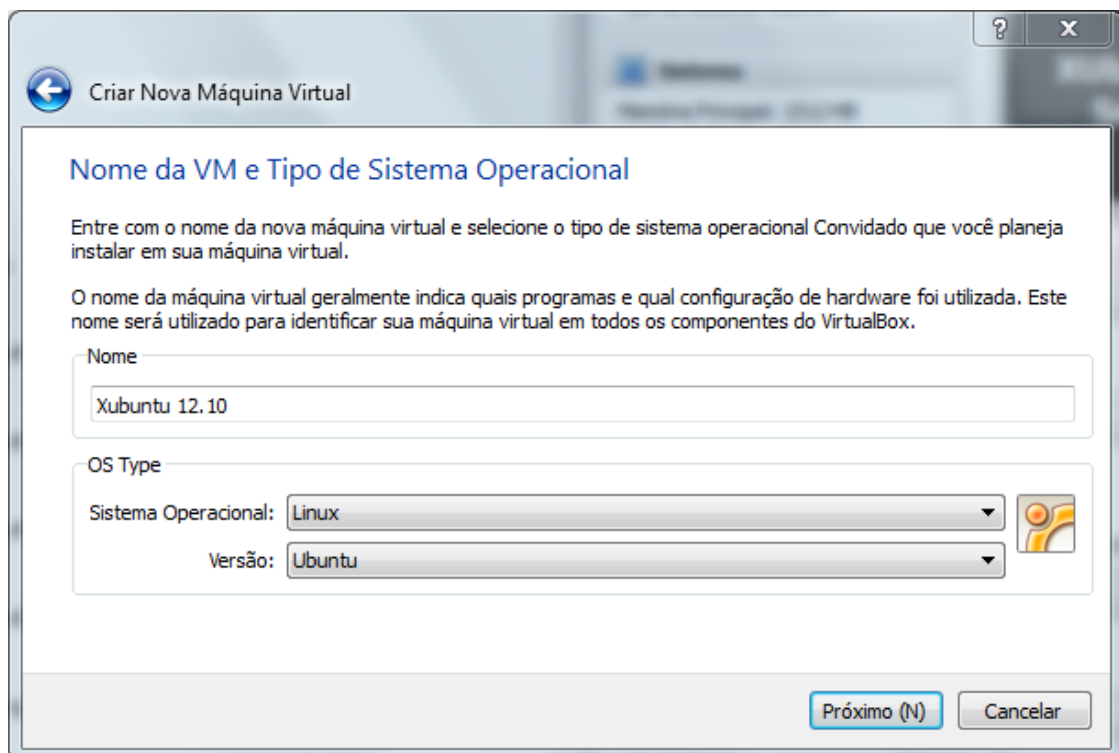



Figura A.1 – Criação da VM – Descrição e SO utilizado na VM
Fonte : Elaborado pelo autor, 2013

O tipo de sistema operacional utilizado surgirá assim que digitarmos o título Xubuntu, sendo este o Linux versão Ubuntu, como mostra a figura A.1.

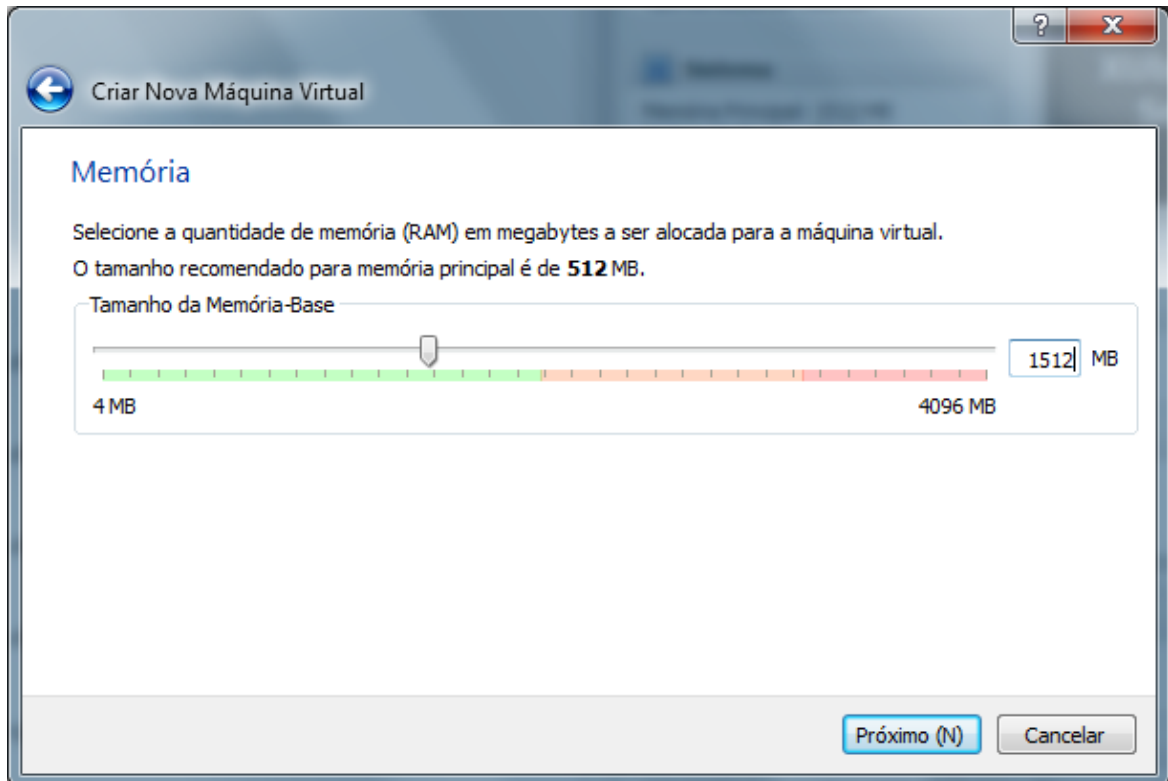



Figura A.2 – Memória reservada para VM
Fonte : Elaborado pelo autor, 2013

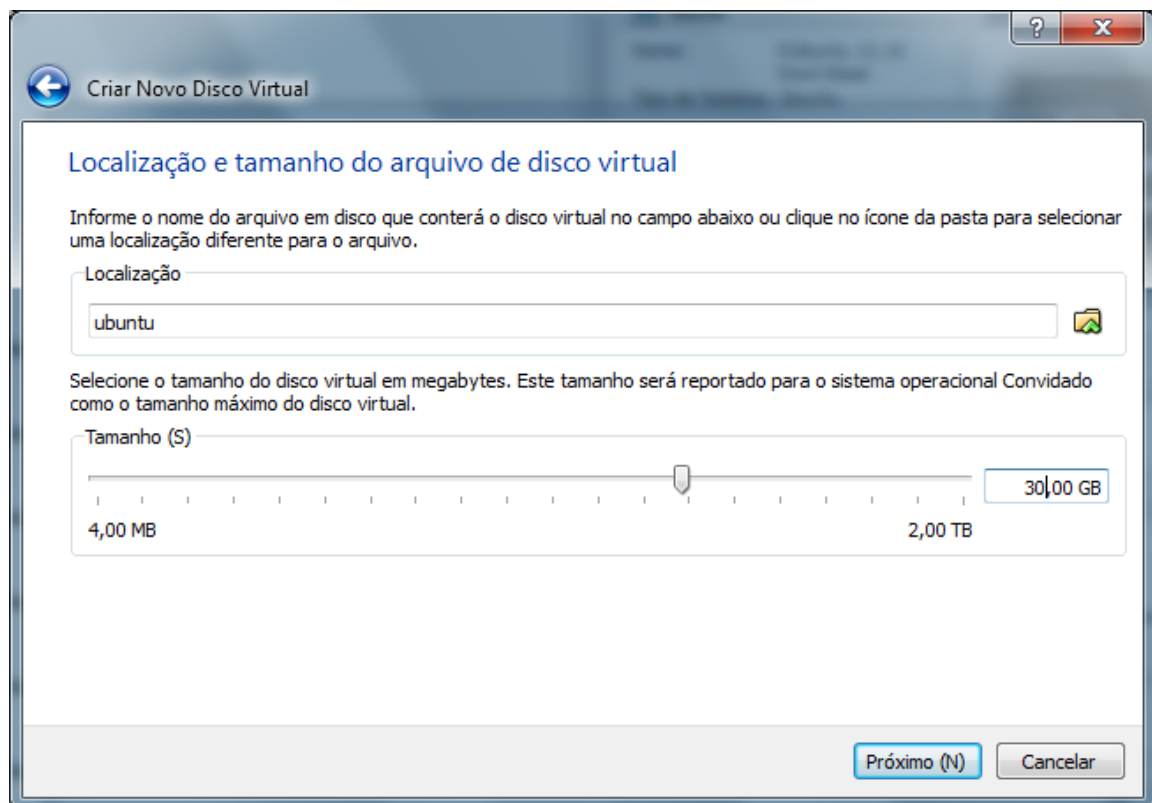



Figura A.3 – Espaço de HD reservado para VM
Fonte : Elaborado pelo autor, 2013

Como figura A.3 mostra que foi selecionado um espaço em HD de 30GB para a VM Servidor de Segurança, porém para as demais VMs o espaço sugerido na instalação (8,00 GB) será o suficiente, já que somente os aplicativos injetores serão instalados.

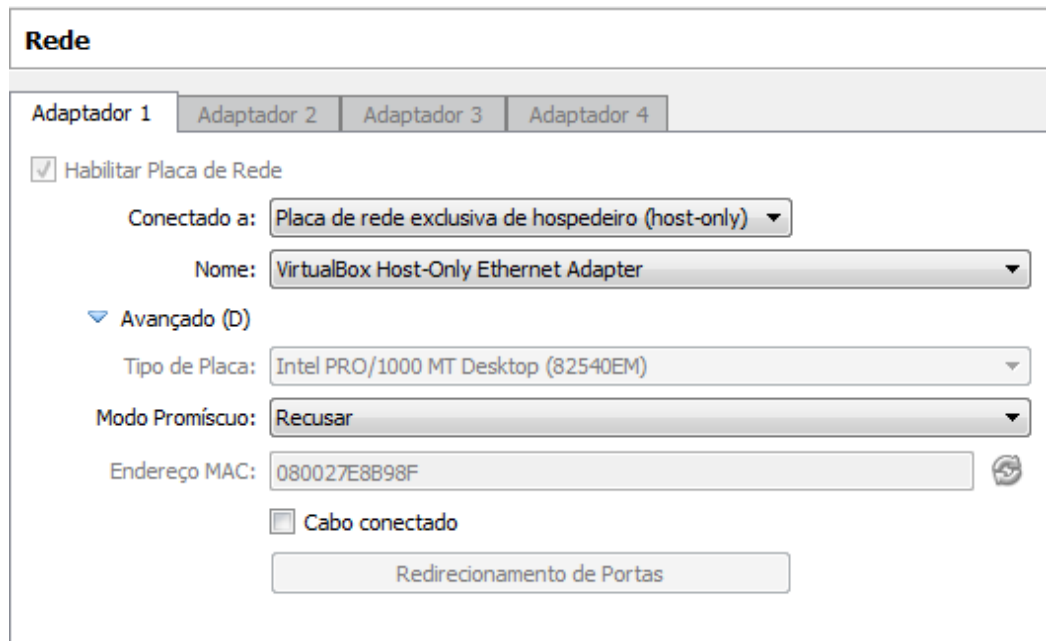



Figura A.4 – Adaptador de rede da VM
Fonte : Elaborado pelo autor, 2013.

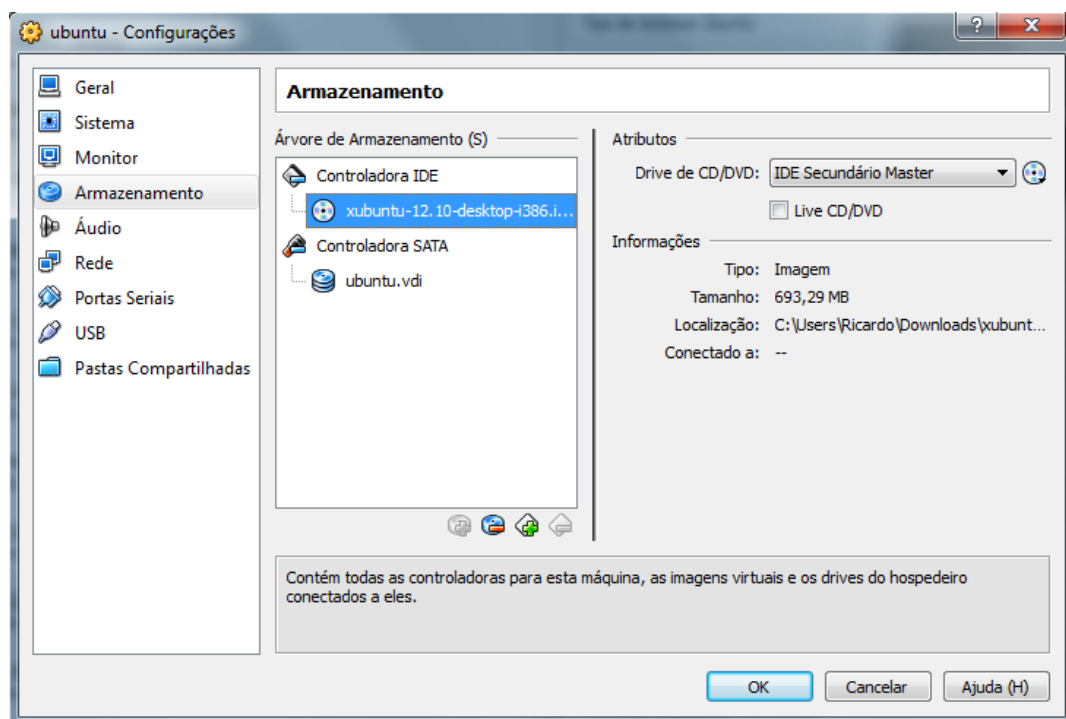



Figura A.5 – Selecionando a ISO contendo a instalação do SO.
Fonte : Elaborado pelo autor, 2013.

Como mostrado na figura A.4, no item **configurações** da VM, opção **rede**, indicamos o tipo de conexão utilizada nesta interface.

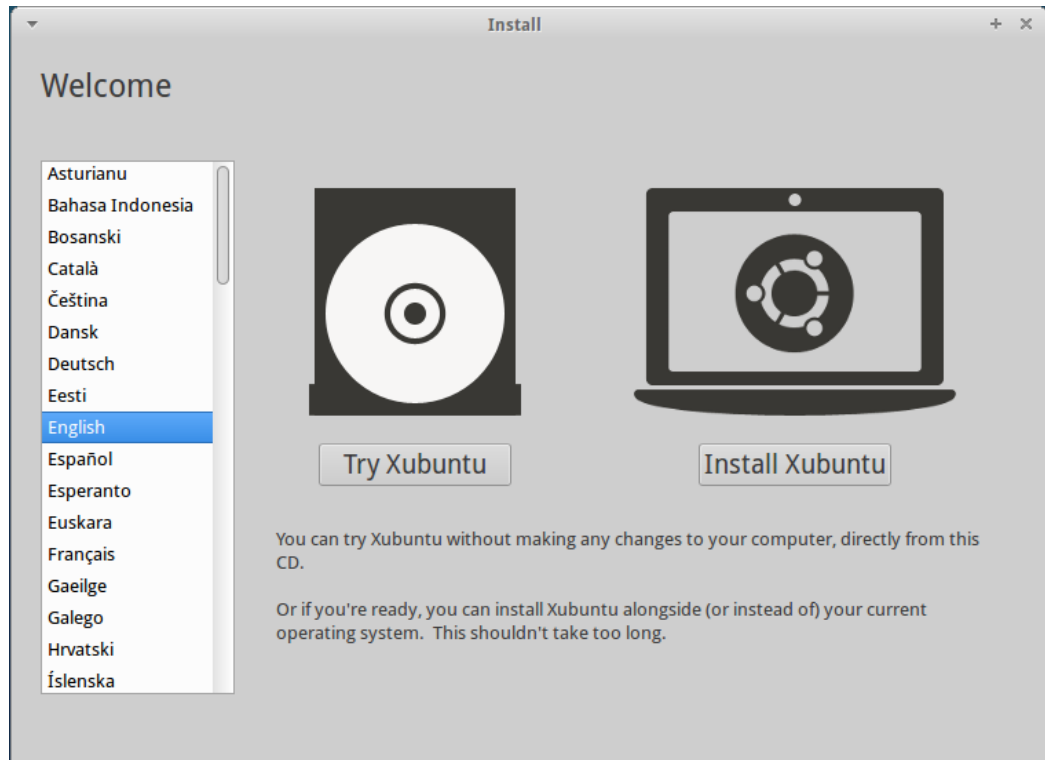



Figura A.6 – Tela de instalação do XUbuntu 12.10
Fonte : Elaborado pelo autor, 2013.

Assim como mostra a figura A.5, no item **configurações** da VM, opção **armazenamento**, indicamos onde está o arquivo de instalação do SO. Este pode estar em DVD-ROM ou em imagem ISO. A instalação do XUbuntu pode ser encontrada no site www.ubuntu-br.org.

Ao se executar a VM criada, o processo de instalação e para concluí-lo basta aceitarmos as opções padrão oferecidas durante o processo, sem necessidade de demais configurações.

Instalando aplicativos e ferramentas

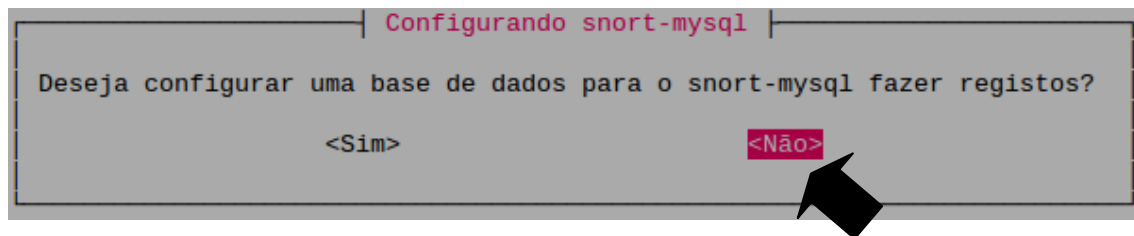
Como mencionado anteriormente, neste trabalho o Snort irá funcionar em conjunto com outros aplicativos afim de proporcionar-lhe as funcionalidades necessárias para atingirmos nossos objetivos. Entre eles estão o banco de dados MySQL, o servidor Web Apache, a linguagem PHP5 com bibliotecas gráficas e o

Snort propriamente dito.

```
#sudo apt-get install mysql-server
#sudo apt-get install apache2
#sudo apt-get install php5
#sudo apt-get install php5-mysql
#sudo apt-get install php5-gd
#sudo apt-get install php-pear
#sudo apt-get install snort
#sudo apt-get install snort-mysql
```

```
Configurando snort-mysql

Deseja configurar uma base de dados para o snort-mysql fazer registos?

<Sim>                    <Não>
```

Baixar e descompactar adodb :

http://prdownloads.sourceforge.net/adodb/

```
#cp /home/ricardo/Downloads/adodb518a.tgz /var/www
#cd /var/www
#tar -xvzf adodb518a.tgz
#rm -f adodb518a.tgz
```

Configuração do MySQL :

```
#mysql
>set password for root@localhost=password('pwmysql');
>create database snort_db;
>grant insert,select on root.* to snort@localhost;
>set password for snort@localhost=password('pwsnort');
>grant create,insert,select,delete,update on snort_db.* to
snort@localhost;
>grant create,insert,select,delete,update on snort_db.* to
snort;
```

```
>exit;
#gzip -d /usr/share/doc/snort-mysql/create_mysql.gz
#mysql -u root -p < /usr/share/doc/snort-mysql/create_mysql
snort_db
```

Configuração do Snort :

```
#sudo nano /etc/snort/snort.conf
ipvar HOME_NET any
ipvar EXTERNAL_NET any
var RULE_PATH /etc/snort/rules
output database : log, mysql, user=snort password=pwsnort
dbname=snort_db host=localhost
```

Executando o Snort :

Modo sniffer :

```
#snort -vde
```

Modo detecção de intrusão

```
#snort -vde -c /etc/snort/snort.conf
```

***Basic Analysis and Security Engine* (BASE) :**

http://prdownloads.sourceforge.net/secureideas/

```
#cp /home/ricardo/Downloads/base-1.4.5.tar.gz /var/www
#cd /var/www
#tar -xvzf base-1.4.5.tar.gz
#rm -f base-1.4.5.tar.gz
#cd base-1.4.5
#cp base_conf.php.dist base_conf.php
#sudo nano base_conf.php
```

Alterar parâmetros no arquivo :

```
$BASE_urlpath    = "/base-1.4.5";
```

```
$DBlib_path       = "/var/www/adodb5";
$DBtype           = "mysql";
$alert_dbname     = "snort_db";
$alert_host       = "localhost";
$alert_port       = "";
$alert_user       = "snort";
$alert_password = "pwsnort";
```

Instalar bibliotecas auxiliares para geração de gráficos :

```
#pear config-set preferred_state alpha
#pear install Image_Color
#pear install Image_Canvas
#pear install Image_Graph
```

Executar no navegador : http://localhost/base-1.4.5

clicar na opção ***setup page***

clicar botão **Create BASE AG**

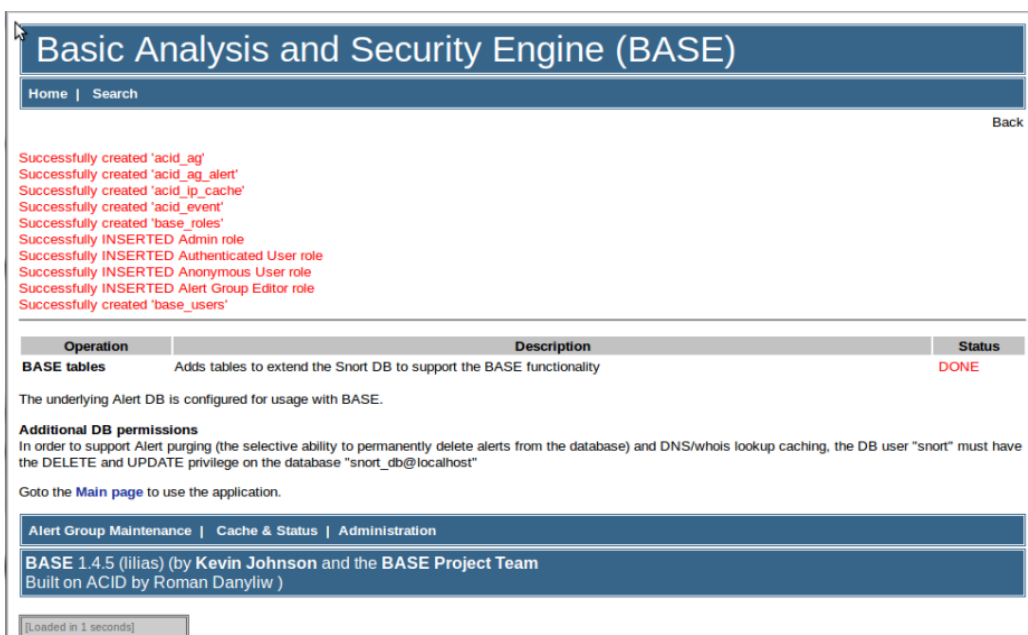



Figura A.7 – Tela inicial do BASE
Fonte : Elaborado pelo autor, 2013.

Instalando o Guardian :

http://www.chaotic.org/guardian/

```
#mv /home/Ricardo/Downloads/guardian-1.7.tar.gz /usr/src
#cd /usr/src
#tar -xvzf guardian-1.7.tar.gz
#cd guardian-1.7
#cd scripts
#cp iptables_block.sh /usr/bin/guardian_block.sh
#cp iptables_unblock.sh /usr/bin/guardian_unblock.sh
#chmod 755 /usr/bin/guardian_block.sh
/usr/bin/guardian_unblock.sh
#cd /usr/src/guardian-1.7
#cp guardian.pl /usr/bin
#chmod 755 /usr/bin/guardian.pl
#cp guardian.conf /etc/
#sudo nano /etc/guardian.conf
```

Alterar itens de configuração do arquivo:

```
HostIpAddr <IP Frente com Intenet>
Interface      eth0
AlertFile /var/log/snort/alert
TargetFile /etc/snort/guardian.target
TimeLimit 86400
```

Criar os arquivos :

```
#touch /var/log/guardian.log
#touch /etc/guardian.ignore
#touch /etc/snort/guardian.target
```

Editar arquivo :

```
#nano /etc/snort/guardian.target
 <IP a ser monitorado>
```

Iniciar o Guardian :

```
#/usr/src/guargian-1.7/guardian.pl -c /etc/guardian.conf
```

Usando logs do snort :

Quando se configura o snort para gravar os alertas no MySQL, os arquivos de log deixam de ser gerados. Logo após a configuração do MySQL no arquivo de configuração do *Snort* (/etc/snort/snort.conf), deve-se adicionar a linha de configuração abaixo para que os arquivos de log de alerta do snort sejam também gerados :

```
output database: log, mysql, user=snort password=123456
dbname=snort host=localhost
output alert_full: /var/log/snort/alert
```

Para parar o serviço do guardian :

```
#kill -9 $(pgrep guardian.pl)
```

Instalação do oinkmaster

http://oinkmaster.sourceforge.net/downloads.shtml

```
#mv /home/Ricardo/Downloads/oinkmaster-2.0.tar.gz /usr/src
#cd /usr/src
#tar -xvzf oinkmaster-2.0.tar.gz
#cd oinkmaster-2.0
#cp /usr/src/oinkmaster-2.0/oinkmaster.pl /usr/bin
#cp /usr/src/oinkmaster-2.0/oinkmaster.conf /etc/
```

Inserir no arquivo /etc/oinkmaster :

```
url = file:///home/ricardo/Downloads/snortrules-snapshot-
2930.tar.gz
```

Iniciar o oinkmaster usando opção por arquivo baixado :

```
#oinkmaster -c -o /etc/snort/rules
```